UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

Diseño de un sistema de control remoto para drones industriales

María Garrido Rodríguez
Tutor: Lluís Gifre Renom
Ponente: Sergio López Buedo

Julio 2020

Diseño de un sistema de control remoto para drones industriales

María Garrido Rodríguez
Tutor: Lluís Gifre Renom
Ponente: Sergio López Buedo

Departamento de Tecnología Electrónica y de las Comunicaciones

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio 2020

Resumen (castellano)

En los últimos años, se ha incrementado de manera significativa la utilización de los drones y en la actualidad empezamos a verlos en nuestra rutina diaria. Su reducido tamaño y relativa facilidad de uso para los usuarios despiertan el interés por este tipo de tecnología. Podemos encontrar drones que se utilizan como juguetes, en trabajos militares, en entrega de paquetes o incluso para grabar eventos desde el cielo. Los drones también se pueden encargar de hacer nuestra vida más segura ya que podemos utilizarlos en situaciones peligrosas para los seres humanos, como por ejemplo en rescates, en situaciones de emergencia no accesibles para una persona o en presencia de elementos tóxicos o peligrosos.

Con esta motivación, este trabajo fin de grado expondrá un proyecto que trata de desarrollar un enlace de comunicación entre el sistema de control de a bordo del dron, y un ordenador en tierra que actuará como estación base.

El desarrollo de este enlace viene motivado por poder integrar todos los canales de comunicación del dron en uno solo basado en protocolos como TCP/IP, lo que permite el uso de radio-enlaces basados en WiFi para entornos controlados, por ejemplo una fábrica, o adaptadores 4G en entornos donde se requiere un alcance mayor y donde no existe cobertura de redes WiFi.

Para lograr estos objetivos será necesario montar una cámara que permita al operador visualizar el entorno, y mandar las órdenes de control adecuadas, por lo que será en lo que nos centremos en este proyecto.

En primer lugar, se hará una prueba de concepto con un micro-ordenador de abordo implementado sobre una Raspberry Pi, una controladora de vuelo estándar y la estación base, de modo que podamos saber si es viable el proyecto que se quiere realizar.

Seguidamente se procederá a cambiar la Raspberry Pi por una placa industrial que nos permita aportar estabilidad al sistema y añadir mayor fiabilidad para un sistema industrial.

De esta manera los siguientes pasos del proyecto serán configurar el nuevo sistema completo y, por último, se llevarán a cabo una serie de pruebas que nos ayuden a comprobar que efectivamente el desarrollo implementado funciona correctamente.

Abstract (English)

In the past few years, the use of drones has increased significantly. They are useful due to their reduced size and the ease of use, making us feel more interested in this kind of technology. In fact, we use them in our day to day as toys, at military work, to deliver packages or even to record videos from the sky.

Drones are also used to make our lives safer in dangerous situations, such as human rescues or emergency situations that are not accessible because of the presence of toxic elements.

The aim of this project is to present and develop a way to communicate the on-board control system of a drone and a ground computer that will work as a base station. The development of this link has been motivated in order to integrate all the drone's communication channels into one based on protocols such as TCP / IP, allowing the use of radio links in WiFi for controlled environments, for example a factory, or 4G adapters in environments where a greater range is required and where there is no coverage of WiFi networks. To achieve our objectives, it would be necessary to develop a camera that allows the operator to visualize the environment and send appropriate control orders; this will be the main focus of the project.

Firstly, we are going to carry out a proof of concept with an on-board micro-computer implemented on a Raspberry Pi, a standard flight controller, and the base station, so that we can know if our project is viable.

Secondly, we will proceed to change the raspberry for an industrial board that allows us to provide stability to the system and add greater reliability for an industrial system. In this way, the next steps of the project will be to configure the new complete system and, finally, a series of tests will be carried out to help us verify that the implemented development is indeed working correctly.

Palabras clave (castellano)

Dron, controladora de vuelo, control remoto, MavLink

Keywords (inglés)

Drone, Flight controller, Remote control, MavLink

Agradecimientos

En primer lugar, dar las gracias a Lluís por saber guiarme en este camino, por las duras tutorías durante horas y seguir disponible incluso en la distancia.

A Visiona Ingeniería de Proyectos, por iniciar este proyecto conmigo y por ser el soporte durante todo este tiempo.

A mis padres, por animarme a elegir este camino y apoyarme en todas las decisiones que he tomado, porque sin ellos no estaría aquí.

A Patricia, mi hermana, por saber entenderme en los momentos duros y ser un pilar fundamental en mi vida.

A Iván, por aparecer en el momento adecuado, por ser luz en el camino, y palabras de apoyo cuando las fuerzas fallaban.

Y, por supuesto, a todos los amigos que me llevo de estos años, porque sin duda habéis hecho esta andadura más fácil y divertida. Cerramos una etapa que no es más que el inicio de muchas otras que espero celebrar a vuestro lado.

Gracias.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

En esta sección motivaremos este trabajo final de grado, además de explicar sus objetivos y presentaremos la organización del proyecto y de la propia memoria.

1.1 Motivación

Los Vehículo Aéreo No Tripulado (VANT) o drones, cuentan cada vez con más presencia entre nosotros y sus aplicaciones son muy variadas. Hoy día, existen distintas soluciones para la conexión entre la estación base en tierra y el dron, pero en todas ellas, las instrucciones de control, la telemetría y el vídeo que el dron recoge y transmite van por canales separados. El manejo del dron, así como la recepción de telemetría se transmiten a través de un radio enlace con el mando, mientras que el vídeo que este graba se transmite por WiFi o 4G hasta un receptor externo, normalmente un teléfono móvil por su manejabilidad, que emplea la misma tecnología. Existen soluciones basadas en controladoras de vuelo que pueden combinarse con pequeños ordenadores, para formar sistemas de control de vuelo sofisticados. Sin embargo, el problema de estas soluciones radica en su baja fiabilidad y corto alcance de sus radioenlaces, tanto entre el mando y el dron, como en el caso de los adaptadores WiFi integrados para transmitir la señal de vídeo. Además, requiere de procesos de codificación de vídeo que deben ejecutarse en los mismos microordenadores, lo cual suele ser un problema computacionalmente complejo que puede interferir con otras tareas críticas relacionadas con el control de vuelo.

Este proyecto viene motivado por poder integrar todos los canales de comunicación del dron en uno solo basado en protocolos como TCP/IP, lo que permite el uso de radio-enlaces basados en WiFi para entornos controlados, por ejemplo una fábrica, o adaptadores 4G en entornos donde se requiere un alcance mayor y donde no existe cobertura de redes WiFi. En este proyecto nos centraremos en desarrollar un canal de comunicación entre el dron y una estación base en tierra que encapsule tanto los comandos de manejo del dron, como la recepción de telemetría de este, así como la recepción de la señal de vídeo que esta captura en tiempo real. En este proyecto, por ser un primer paso a la integración de los canales de comunicación en un solo radio enlace, hemos decidido empezar por el caso de WiFi. Para ello, primero se desplegará una solución completa basada en una Raspberry Pi tal como documenta el propio fabricante de la controladora de vuelo que emplearemos, ArduPilot PixHawk, para aprender a usar la controladora y la transmisión de vídeo. En segundo lugar, se pretende investigar posibles computadores industriales miniaturizados que puedan sustituir a la Raspberry Pi para mejorar la estabilidad del sistema.

1.2 Objetivos

El objetivo principal de este proyecto será desarrollar el radio enlace para el VANT, para ello se considerarán los siguientes sub-objetivos:

En primer lugar, se realizará una prueba de concepto con una Raspberry Pi 3 y la controladora PixHawk 2 para conseguir una primera solución que nos permita comunicar el VANT con la estación base en tierra.

En segundo lugar, se hará una comparativa de soluciones tecnológicas disponibles con la finalidad de reemplazar la Raspberry Pi 3 utilizada en la prueba de concepto por una placa industrial que nos ofrezca mayor estabilidad. Una vez seleccionada la nueva placa, nos enfocaremos en tratar de portar la comunicación con la controladora y la estación base desplegando el sistema de a bordo sobre el microordenador seleccionado.

Por último, se estudiarán las posibilidades para transportar las señales serie de control del vuelo mediante un radio-enlace WiFi integrado con la retransmisión de la señal de video.

1.3 Fases del proyecto

En base a los objetivos fijados en la sección 1.2, se organizará el proyecto en diferentes fases que quedan reflejadas en la Tabla 1.

Tabla 1: índice de tareas

	Tarea	Horas
T1	Investigación del estado del arte	20
T2	Desarrollo con Raspberry Pi 3	40
T3	Comparativa de soluciones tecnológicas disponibles	70
T3.1	Elección de placa	50
T3.2	Elección del dispositivo WiFi	20
T4	Desarrollo del sistema de a bordo	150
T4.1	Configuración placa Toradex	80
T4.2	Desarrollo con placa Toradex	70
T5	Agregación del Sistema WiFi	220
T5.1	Compilación del controlador del dispositivo WiFi	110
T5.2	Compilación del núcleo del sistema operativo de la placa	110
T6	Realización de pruebas	15
T7	Preparación de la memoria	220
T8	Preparación de la presentación y la memoria	20

Adicionalmente, en la Figura 1-1, vemos la representacion de esas tareas mediante un diagrama de Gantt. Como vemos casi todas las tareas son dependientes, lo que implicará que si alguna se retrasa esto afecte a todo el proyecto por no poder paralelizar las tareas principales.

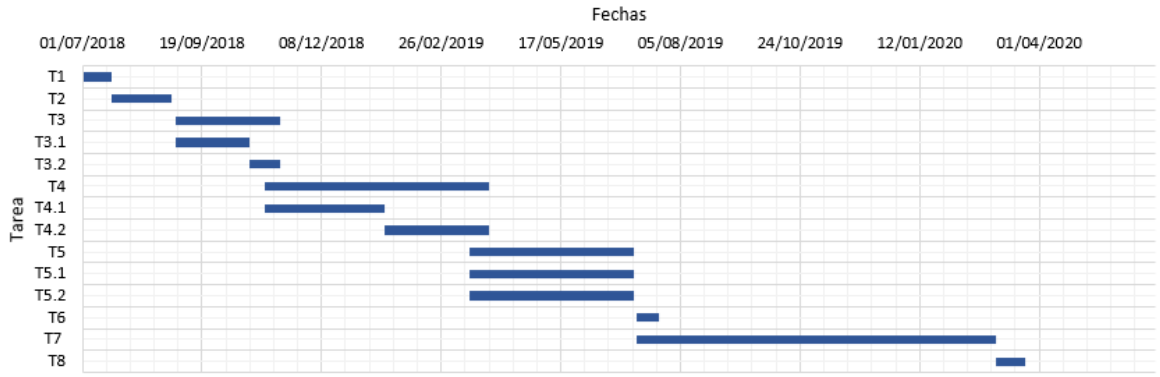

Figura 1-1 Diagrama de Gantt

1.4 Organización de la memoria

La memoria de este trabajo final de grado se ha organizado de la siguiente forma:

- En el capítulo 2, titulado **¡Error! No se encuentra el origen de la referencia.**, se introducen las tecnologías relevantes que se han empleado en el desarrollo de este trabajo.
- En el capítulo 3, titulado Diseño y desarrollo, se exponen las elecciones para el diseño del proyecto, así como el desarrollo de este.

- En el capítulo 4, titulado Integración, pruebas y resultados, se presentan las pruebas realizadas en cada una de las fases del proyecto para ver que la solución obtenida funciona correctamente.
- Finalmente, en el capítulo 5, titulado Conclusiones y trabajo futuro, se resumen las conclusiones más relevantes de este trabajo y se plantean algunas líneas de trabajo e investigación futuras que pueden dar continuidad a este trabajo.

2 Estado del arte

En esta sección, se presentan las tecnologías más relevantes para el desarrollo de este trabajo final de grado. En particular nos centraremos en describir la tecnología WiFi, los aspectos fundamentales de un dron tales como qué es y cuáles son sus componentes, cuáles son las funciones que pueden llevar a cabo, una descripción de los sistemas de control en tierra, los equipos controladores de vuelo de a bordo y el protocolo de comunicación utilizado.

2.1 Tecnología WiFi

WiFi es una tecnología que nos permite conectar de forma inalámbrica varios dispositivos [1] y, aunque actualmente existen muchas familias retro-compatibles entre ellas de normas inalámbricas dentro de esta tecnología, nos centraremos en hablar de los protocolos IEEE 802.11n [2] y IEEE 802.11ac [3] ya que en este proyecto utilizaremos estos dos protocolos al hablar de WiFi a 2.4 GHz o WiFi a 5 GHz. Para ello, pasaremos en primer lugar a compararlos.

El estándar IEEE 802.11n, aprobado después de su ratificación en septiembre de 2009, utiliza la tecnología Multiple-input Multiple-output (MIMO), en español Múltiple entrada múltiple salida, que es la que permite la utilización de múltiples antenas para la aceleración de la transmisión de datos, lo que permite aumentar la velocidad de transferencia de datos de la red con respecto a sus antecesores 802.11a y 802.11g.

El estándar IEEE 802.11ac, aprobado en julio de 2014, utiliza la tecnología MIMO multi-usuario (Multi-User MIMO, MU-MIMO) que permite la asignación de flujos espaciales a más de un dispositivo en el mismo tiempo. La cantidad de dispositivos depende del número de antenas y del modo de implementación de MU-MIMO, aumentando la velocidad de transferencia en todos los casos.

La norma IEEE 802.11ac es una mejora de IEEE 802.11n, que se ha basado principalmente en mejorar tanto la tasa de transferencia como el ancho de banda disponible. Por ello, las principales diferencias entre trabajar a 2.4 GHz o a 5 GHz son la cobertura que ofrecen y la velocidad a la que podemos transmitir. Para el caso de 2.4 GHz podremos transmitir a distancias más largas, pero perderemos velocidad, y en el caso de 5 GHz justo al revés, cubriremos menos distancia, pero ganaremos en velocidad, hasta 1.3Gbps frente a 600 Mbps. Además, cuando utilizamos la frecuencia de 5 GHz nos encontraremos que la red está menos utilizada, debido a su novedad, ya que menos dispositivos trabajan a esta frecuencia y por lo tanto tendremos menos interferencias y problemas de conexión.

2.2 Vehículo aéreo no tripulado (VANT)

Un dron es un VANT, por lo que el piloto opera en remoto, o es un sistema totalmente autónomo. Pilotar un VANT no es demasiado complicado, especialmente los que se utilizan en la vida diaria como juguete, por ello se está produciendo un crecimiento exponencial de este tipo de dispositivos. Un dron nos permite tanto movimientos como giros en tres dimensiones; existen 4 tipos de movimientos y giros:

- Deslizar: se trata de mover el dron hacia la izquierda o hacia la derecha
- Cabecear (pitch): sirve para inclinar el frontal del dron y por lo tanto conseguir que se mueva hacia delante o hacia atrás
- Derrapar (yaw): se utiliza para hacer girar el dron sobre sí mismo y puede ser tanto en sentido horario como antihorario

- Acelerar (throttle): sirve para mantener el dron en el aire y hacer que ascienda o descienda.

Sin embargo, actualmente se utilizan VANT para mucho más aparte de como un entretenimiento y a continuación vamos a pasar a describir algunas aplicaciones:

- Grabación/retransmisión/videovigilancia de eventos y personas en tiempo real [4]: Permiten obtener perspectivas no disponibles mediante técnicas de grabación clásicas o fijas. También permiten el seguimiento de criminales y/o sospechosos en tiempo real. El uso de drones para grabación en tiempo real también está muy extendido en el control de fronteras a nivel militar e incluso en funciones de espionaje.
- Entrega de paquetes [5]: Las nuevas tendencias de reparto apuntan a que la paquetería del futuro se llevará a cabo mediante drones, cuando la legislación, y características físicas de los paquetes (peso y dimensiones) respecto del dron lo permitan.
- Acción en caso de emergencia y/o seguridad [6]: dado su reducido tamaño y maniobrabilidad, se pueden emplear drones en casos de desastre para proporcionar a las víctimas pequeños paquetes con material de cura y/o auxilio tales como medicamentos, cuerdas, arneses, etc.
- Debido a que los drones pueden incorporar diferentes sensores para analizar el entorno, se pueden emplear en la detección de incendios forestales [7] empleando sensores de temperatura y cámaras infrarrojas, o la detección de escapes de productos nocivos tales como gases u otros productos químicos.

2.2.1 Componentes de un VANT

Los componentes básicos que conforman el sistema de control y vuelo de un dron se ilustran en la Figura 2-1 y se describen a continuación:

1. Controladora de vuelo: Se trata del corazón del sistema de vuelo (formada por los sensores 2, 3, 4 y 5), que se utilizará para comunicarse con el ordenador de a bordo del VANT y este transmitirá los datos a la estación en base tierra. Esta controladora nos permitirá calibrar algunos de los sensores del VANT, mediante la aplicación que se ejecuta en la estación base en tierra.
2. Rotor: En un dron se utilizan motores sin escobillas (en inglés, brushless) en vez de motores clásicos, ya que los clásicos tienen menor eficiencia y se deterioran debido al rozamiento de las escobillas y los terminales. Los motores brushless invierten la estructura del motor y mantienen los electroimanes fijos en la parte exterior del motor y el rotor interior se compone de imanes que rotan acorde a las polaridades de los electroimanes exteriores. Entonces, es necesario tener circuitos electrónicos que controlen los cambios de polaridad de los electroimanes para generar los campos magnéticos apropiados que hagan girar el rotor. Estos circuitos son los ESC.
3. Circuito electrónico de control de velocidad ESC (del inglés Electronic Speed Controller): cada uno de los ESC (uno por cada rotor) se encargará del control de la velocidad de giro de su rotor asociado. Normalmente emplean una señal de control basada en la modulación de anchura de pulso (PWM, del inglés Pulse Width Modulation) para regular la velocidad de giro del rotor proporcionalmente al ciclo de trabajo del pulso modulado en anchura.
4. Acelerómetro: Es el sensor encargado de medir la aceleración del VANT cuando este se desplaza. Permite medir los cambios de velocidad originados externamente durante el vuelo del dron y se puede utilizar para medir la orientación utilizando la

gravedad terrestre, por ejemplo, detectar cuando el dron es impactado por una ráfaga de viento, y aplicar las correcciones apropiadas.

5. Giroscopio: Es el sensor que permite medir la orientación del dron. El funcionamiento de este elemento, a diferencia del acelerómetro, es independiente de la aceleración y gravedad terrestre, por lo que puede operar en el espacio exterior y proporciona información precisa en caso de tener que realizar piruetas o movimientos rápidos de giro sobre sí mismo cuando el dron está en pleno vuelo.
6. Barómetro: Es el sensor encargado de medir la presión atmosférica que se aplica al dron. Con ello, el dron puede conocer su altura respecto de tierra, una información extremadamente importante para las etapas de despegue y aterrizaje.
7. Magnetómetro: Es el sensor que permite medir la fuerza del campo magnético terrestre, por lo que proporciona información sobre la orientación del dron respecto de los polos terrestres.
8. Batería: Es el componente encargado de proporcionar la energía al VANT. Este dispositivo puede ser desde una simple batería hasta unos pequeños paneles solares que se cargan mediante laser [8], y aunque normalmente es uno de los elementos más pesados del dron también permite prolongar su autonomía de vuelo.
9. Radio enlace con la estación base en tierra: es el módulo que posibilita que el VANT se comunique con la estación base, ya sea a través de WiFi, Bluetooth, módems 3/4G, etc. Junto con la autonomía que aporta la batería, condicionan la distancia máxima a la que puede alejarse el dron.
10. GPS: Sensor que, empleando satélites de posicionamiento global (Global Positioning System, GPS) proporcionará las medidas de geolocalización al dron para que pueda conocer su posición, se trata de un sensor opcional que podemos utilizar por ejemplo si queremos que el dron sea capaz de seguir rutas preprogramadas y pueda volver a la estación base.
11. Ordenador de a bordo: Se trata de una pequeña computadora que se encargará de facilitar los datos de vuelo del dron, obtenidos a través de la controladora de vuelo, como la telemetría, y proporcionar los comandos de vuelo o los datos de la ruta que se debe seguir.

En este caso se utilizará la controladora ArduPilot Pixhawk 2 [9], principalmente porque Pixhawk es el estándar de hardware para los pilotos automáticos de código abierto. La controladora Pixhawk 2 es un piloto automático flexible destinado principalmente a fabricantes de sistemas comerciales. Se basa en el diseño de hardware abierto Pixhawk-Project FMUv3 y ejecuta PX4 en el sistema operativo Nutt.

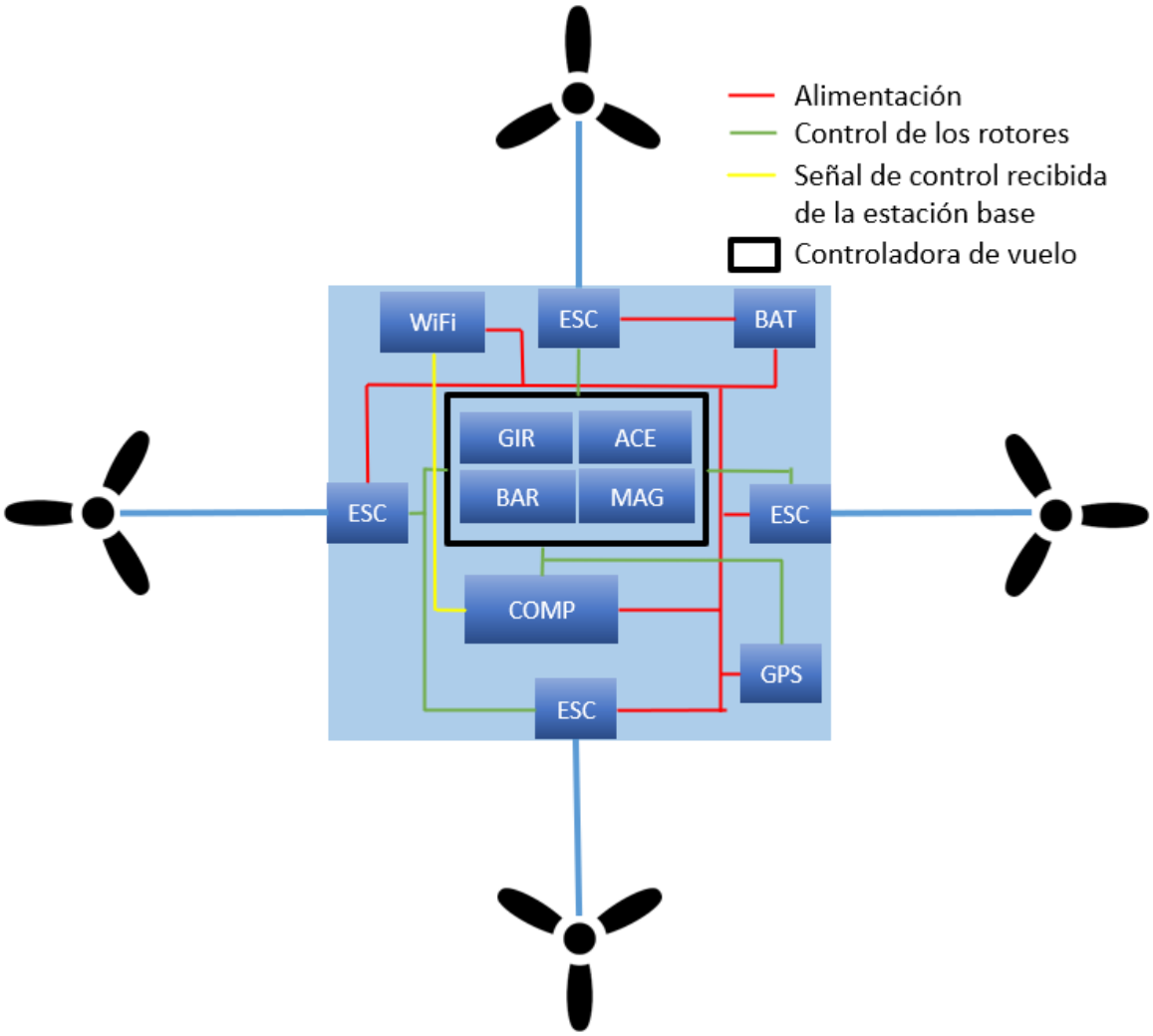

Figura 2-1:Esquema de los componentes de un VANT

2.3 Sistema de control en tierra

Para ser capaces de manejar el dron desde tierra, necesitaremos un sistema de control que nos permita, no solo enviarle comandos de control, sino también conocer el estado del VANT en todo momento para poder actuar en caso de que sea necesario. En general, nos interesará recibir tanto telemetría como video en tiempo real, también nos interesará poder realizar tareas de mantenimiento del dron como aplicar actualizaciones o calibrar sus sensores.

Para asegurar un correcto funcionamiento, el primer paso a realizar con un dron será calibrarlo, esta operación se realiza, normalmente, conectando la controladora de vuelo al ordenador que hará de base y que tiene corriendo el programa de control de vuelo, una vez iniciada la aplicación será capaz de detectar automáticamente el vehículo y pasará a pedirnos que calibremos cada uno de los sensores siguiendo los pasos que aparecen por pantalla. Normalmente, estas operaciones consistirán en posicionar y girar el dron siguiendo las instrucciones del programa.

En esta sección, nos centraremos por tanto en estudiar Mision Planner y QGroundControl que son dos de los programas de vuelo que más utilizados que existen en el mercado actualmente. Se han descartado otras opciones como Free Flight pro o DJI GS Pro porque

son restrictivas y solo tienen aplicación para móvil, y otras como Ground Station porque están descontinuadas, etc.

En primer lugar, debemos hablar de que ambos programas permiten calibrar cada uno de los sensores y además podremos conocer mediante el sistema de logs y errores si alguno de ellos está fallando y por ello no es posible armar el dron. Que no sea posible armar el dron significa que el dron no está listo para volar porque alguna de sus partes no está correctamente calibrada o la señal GPS tiene problemas. Además, en los dos tenemos la opción de crear las rutas que queremos que siga el dron, de manera que pueda ser más autónomo.

1. **Mission Planner [10]:** Esta herramienta nos permite tanto conocer el estado del VANT mientras vuela como controlarlo en tiempo real en vista de primera persona.
2. **QgroundControl [11]:** Esta herramienta cuenta con la configuración necesaria para correr los comandos necesarios si queremos utilizar transmisión de video. Además, soporta Windows, OS X, Linux, iOS y Android, por lo tanto, nos aporta mucha más versatilidad en el dispositivo que queramos utilizar como base en tierra.

En este proyecto, emplearemos la herramienta QGroundControl ya que, aunque ambos nos proporcionan las herramientas que necesitamos para controlar el dron y conocer su estado en todo momento, la estación de control QGroundControl soporta más sistemas operativos.

2.4 Protocolo MAVLink

MAVLink (Micro Air Vehicle Communication protocol) es un protocolo de mensajería para la comunicación con drones y entre componentes de drones integrados [12], es un protocolo que emplea mensajes binarios no basados en texto para mitigar posibles problemas relacionados con las restricciones de ancho de banda y evitar la complejidad computacional que conlleva el procesado de mensajes basados en texto.

En el protocolo MAVLink existe la posibilidad de definir dialectos [13], que son archivos XML que definen familias de mensajes, enumeraciones y comandos específicos del fabricante. Cada dron implementará la familia de mensajes que haya decidido su fabricante. Se utilizan las definiciones de los dialectos para generar bibliotecas MAVLink para cada uno de los fabricantes. Estas bibliotecas serán utilizadas por los drones y por las estaciones de control para poder comunicarse entre ellas.

Las características más importantes de este protocolo son que es muy eficiente, muy fiable, es compatible con muchos lenguajes de programación, habilita las comunicaciones internas entre componentes del dron y externas con las estaciones base y otros drones y permite hasta 255 sistemas concurrentes en la red ya sea de vehículos como de estaciones en tierra.

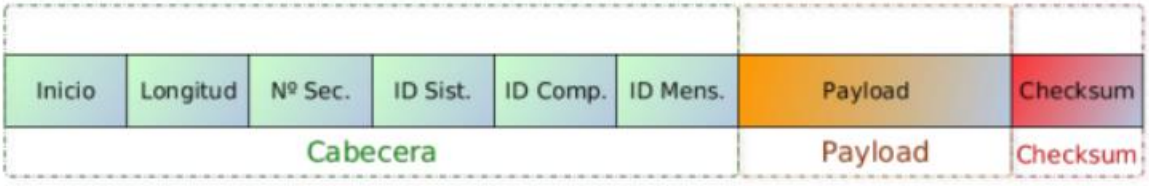

Figura 2-2: Estructura de un paquete MAVLink

El protocolo MAVLink sigue la anterior secuencia codificada de Bytes que se observa en la Figura 2-2 se muestra como el paquete que envía la controladora debe contener la

longitud que va a tener la carga útil del mensaje (payload), el número de secuencia para que seamos capaces de reordenar los paquetes en el destino en caso de llegar en desorden, el identificador del vehículo remitente por si se diera la opción de tener más de un vehículo conectado a la misma red, el identificador del componente remitente para poder diferenciar entre diferentes componentes dentro de un mismo sistema (llamamos componente por ejemplo al piloto automático o a una cámara), el identificador del mensaje que nos indicará como debemos leer el mensaje que se nos proporciona a continuación, se diferencia con el número de secuencia en que este ID solo está asociado a la carga útil del mensaje no a todo el paquete como en el caso del ID de secuencia, y por último el checksum que se utilizará para el control de errores.

2.5 Conclusiones

Este capítulo hemos presentado las tecnologías más relevantes relacionadas con este trabajo. En primer lugar, hemos hablado de la tecnología WiFi a 2.4 GHz y 5 GHz viendo las diferencias entre ellas, para seguir explicando qué es un vehículo aéreo no tripulado o dron, así como cuáles son sus componentes y cómo está diseñado. También se ha presentado los componentes que conforman un sistema de control de vuelo y el protocolo que se utiliza en la comunicación de un dron con la estación base en tierra.

3 Diseño y desarrollo

En este capítulo se documentarán los distintos pasos que se han seguido en este proyecto; el primero de ellos será conseguir que una Raspberry Pi 3 actúe como ordenador de a bordo del VANT, para después llegar a una fase de diseño donde se elija una placa más adecuada para este proyecto y se implemente el enlace de comunicaciones sobre ella.

3.1 Desarrollo con Raspberry Pi

El primer paso ha sido establecer la comunicación entre nuestro equipo de trabajo, la Raspberry Pi 3 que hemos empleado como ordenador de a bordo y la controladora de vuelo Pixhawk 2. Para ello, se ha configurado la Raspberry Pi 3 con su sistema operativo Raspbian [14], que es una distribución de Linux oficial, empleando una tarjeta SD donde se ha almacenado el sistema operativo empleando la herramienta Raspberry Pi Image [15] proporcionada por Raspberry. Dicha tarjeta SD se ha empleado como disco de almacenamiento para la Raspberry Pi.

Seguidamente, conectaremos la Raspberry Pi a una pantalla mediante su puerto HDMI y utilizaremos un teclado USB para configurarla. También conectamos la Raspberry Pi con cable Ethernet a nuestro enrutador de casa para poder actualizar la Raspberry Pi, el primer inicio de Raspberry siempre lo requiere porque las imágenes se regeneran con poca frecuencia. Después, con el menú que se observa en la Figura 3-1 tendremos que configurar la Raspberry y habilitar la comunicación SSH para que podamos conectarnos a la placa sin necesidad de interfaz gráfica en el futuro, que es como trabajaremos nosotros, además de habilitar el puerto serie y la cámara para poder llevar a cabo la comunicación con la controladora y ser capaces de trasmitir video en directo. En este punto también debemos configurar la conexión a Internet inalámbrica, si nos vamos a conectar vía WiFi. Para poder hacer todo esto, debemos acceder desde el escritorio de la Raspberry a preferences menú->Raspberry Pi Configuration. Por último, se debe actualizar el sistema y reiniciar la Raspberry.

El siguiente paso será configurar la controladora de vuelo, Pixhawk 2. Para ello, se debe conectar mediante USB a la estación en tierra en la que esté corriendo la aplicación QGroundControl, de tal forma que esta detecte la controladora.

Si existe una versión nueva del firmware de la Pixhawk 2, QGroundControl nos propondrá actualizarla. Una vez se haya completado el proceso, se pasará a calibrar cada uno de los sensores de la Pixhawk 2 siguiendo las instrucciones que nos indicará QGroundControl que consisten en mover de forma específica la controladora, podemos ver un ejemplo del menú de configuración en la Figura 3-2. Además, se deben configurar los siguientes parámetros manualmente en el menú de configuración que aparece una vez detectado el vehículo:

- `SERIAL2_PROTOCOL = 1` para habilitar el protocolo MAVLink en el puerto serie
- `SERIAL2_BAUD = 115200` que es la velocidad de transmisión a la que se comunicará la controladora con las Raspberry Pi 3.
- `LOG_BACKEND_TYPE = 3` para que ArduPilot almacene los logs en un fichero y a la vez los mande vía protocolo MAVLink.

Tabla 2: Comando de establecimiento de enlace dron-tierra.

```
sudo mavproxy.py --master=/dev/serial0 –baudrate 115200 --out
192.168.1.5:14550 --aircraft MyCopter
```

Por último, se utilizará el comando de la Tabla 2 para enlazar la controladora + Raspberry con la aplicación QGroundControl, siendo `serial0` el puerto de la Raspberry que se está utilizando para la comunicación, `192.168.1.5` es en este caso la dirección IP del PC donde se está ejecutando el programa de vuelo y `14550` el número de puerto para establecer la comunicación. En la sección 4.1 Pruebas con Raspberry Pi 3 se mostrará el resultado de la ejecución de estos comandos, así como las pruebas de transmisión del video.

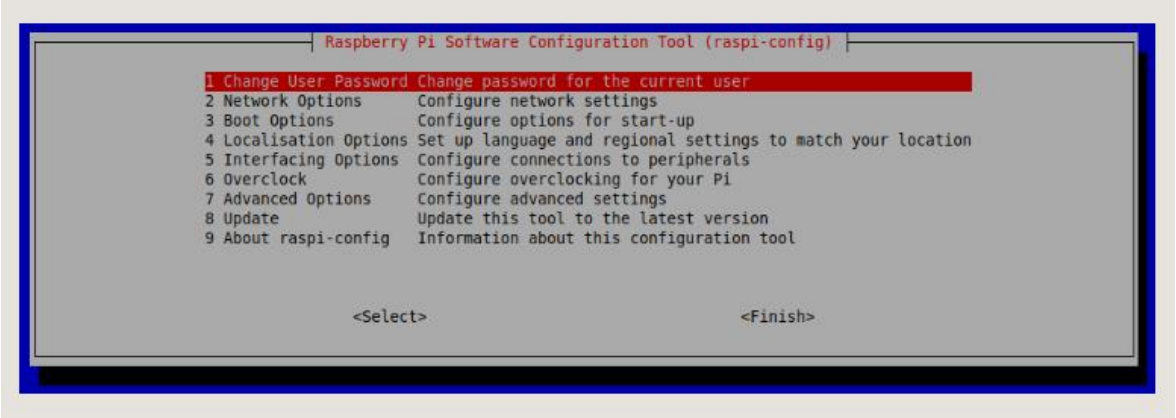

Figura 3-1: configuración de Raspberry Pi 3

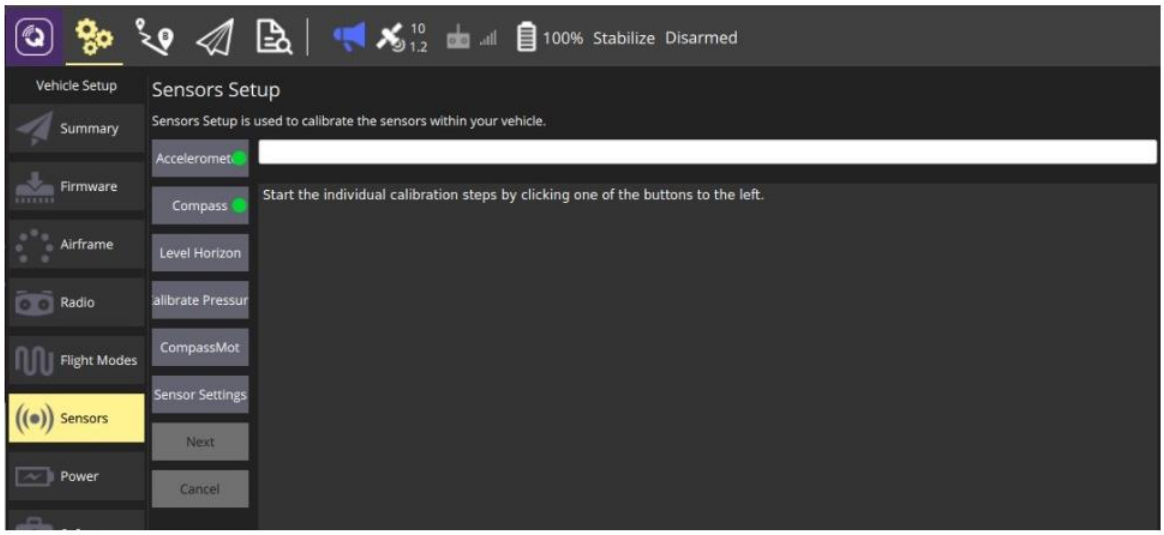

Figura 3-2: Calibración de sensores QGroundControl

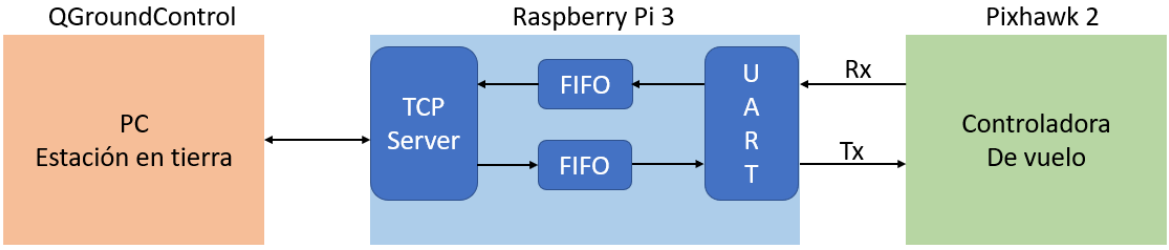

Figura 3-3: Esquema de comunicación

En la Figura 3-3, se ilustra de forma esquematizada cómo sería la conexión entre QGroundControl, el ordenador de a bordo y la controladora de vuelo. En la Raspberry Pi 3 debe correr un proceso que convierta los comandos recibidos de la estación base a través de un servidor TCP a comandos que se mandan a la Pixhawk vía UART, y viceversa. La comunicación entre QGroundControl y Raspberry Pi 3 se realiza mediante un enlace soportado por una conexión TCP/IP que se encargará de transmitir los comandos de la estación base a la controladora de vuelo del dron y recibir la telemetría en sentido contrario, de tal forma que en la estación base se pueda saber en todo momento que está sucediendo con el VANT. La comunicación entre la Raspberry Pi 3 y la controladora de vuelo Pixhawk 2 emplea un enlace serie que emplea el estándar UART.

Una vez se ha conseguido establecer la conexión para la comunicación que enviará la telemetría del vuelo, el segundo paso se trata de conseguir retransmitir el video en tiempo real de la cámara conectada al VANT, en este caso a la Raspberry Pi 3. Para ello se vuelve a utilizar la aplicación QGroundControl ya que tiene la posibilidad de recibir la retransmisión de video a la vez que muestra la ruta que está siguiendo el vehículo.

Tabla 3: Comandos para la retransmisión del video

1	`sudo apt install python`
2	`sudo pip install python-picamera`
3	`raspivid -o - -t 0 -n -w 600 -h 400 -fps 12 | cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8000/}': demux=h264`

Para configurar la retransmisión de video desde la Raspberry Pi 3 debemos instalar el paquete de Python mediante el primer comando de la Tabla 3 y también debemos instalar el módulo de la cámara de Raspberry con el segundo comando de la Tabla 3. Una vez configurada la aplicación se tiene que abrir una terminal que esté conectada por SSH a la Raspberry Pi 3 e introducir el tercer comando de esta tabla que se encargara de iniciar el servidor de retransmisión de video en la Raspberry Pi 3. En este comando 8000 corresponde al puerto que se ha configurado en QGroundControl y “rtsp” es el protocolo que se está utilizando para ello. En la sección 4.1Pruebas con Raspberry Pi 3 se mostrará el resultado de la ejecución de estos comandos, así como las pruebas de transmisión del video.

Luego, en QGroundControl, Para configurar la retransmisión del video se debe ir a la configuración general de la aplicación dedicada a este apartado, e introducir la fuente de video, en este caso se configura “RTSP Video Stream” para recibir el video en directo, por lo que la aplicación necesitará como parámetros adicionales la dirección y puerto RTSP en formato de URL donde el servidor del dron proporciona la señal de video y el campo relación de aspecto (aspect ratio, en inglés), que es la proporción entre ancho y alto de la imagen que se retransmite. En este caso, se utilizará la dirección URL “rtsp://192.168.0.27:8000” y la relación de aspecto que viene por defecto, 1.7777.

Una vez se ha conseguido tanto establecer la conexión de control y telemetría, y la de video en tiempo real, dado que la Raspberry Pi 3 no es un sistema de prestaciones industriales, puede funcionar de forma inestable. Por este motivo, en las siguientes secciones estudiaremos su reemplazo por un mini-ordenador más robusto de prestaciones industriales.

3.2 Selección de placa

En esta sección, se hará un estudio de diferentes placas que existen en el mercado actualmente. Se prestará especial atención a cumplir con las especificaciones siguientes:

- Un mínimo de 2 puertos USB de manera que se pueda conectar tanto el adaptador WiFi como la cámara que se quieren integrar en el futuro.
- Un puerto serie para la comunicación que permita la comunicación con la controladora de vuelo.
- La posibilidad de conexión WiFi a 5 GHz ya que así tendremos una banda de frecuencias más libre como se comenta en el capítulo 2 Estado del arte, reduciendo así las posibles interferencias. Además, nos proporcionaría mayor ancho de banda y velocidad.

En el consumo y el peso del sistema completo ya que estará integrado en un VANT. La sección recogerá el estudio de tres familias de placas, similares a Raspberry en la sección 3.2.1, placas basadas en Arduino en la sección 3.2.2 y placas del fabricante Toradex en la sección 3.2.3.

3.2.1 Placas similares a Raspberry

En primer lugar, se estudiarán las placas que tienen ciertas similitudes con la Raspberry Pi 3 que ya hemos utilizado, ya que simplificaría el desarrollo del proyecto. En la Tabla 4 se resumen las principales especificaciones de las placas consideradas en esta sección.

Tabla 4: Placas similares a Raspberry

Placa	Procesador	RAM	Entrada de video	USB 2.0	USB 3.0	Potencia	UART	WiFi
Orange Pi Plus 2E [16]	H3 Quad-Core Cortex A7	2 GB DDR3	CSI	4xHost, 1xOTG	✗	DC 5V 2A	✓	✓
NanoPi Neo plus 2 [17]	AllWinner H5, Quad-Core 64 bits	1 GB DDR3	✗	2xHost	✗	DC 5V 2A	✓	✓
Odroid XU4 [18]	Samsung Exynos-5422 Octa Core	2 GB DDR3	✗	1xHost	2xHost	DC 5V 4A	✓	✓
UP Core [19]	Intel Atom x5-Z8350 64 bit Quad-Core	2/4 GB DDR3	CSI	2xpin header	1xHost	DC 5V 4A	✓	✓
Odroid C2 [20]	Amlogic ARM Cortex-A53 1.5GHz Quad-Core	2GB DDR3	✗	4xhost	✗	DC 5V 2A	✓	✗
JaguarBoard [21]	Intel Atom Z3735G@1.83 GHz	1 GB DDR3	✗	3	✗	DC 5V 2A	✗	✗
BeagleBone Blue [22]	AM3358x 1GHz ARM Cortex- A8	512 MB DDR3	✗	1xhost	✗	DC 9-18V 4A	✓	✗
ROCK64 4K60P [23]	RK3328 64-bit Quad Core A53	1/2/4 GB DDR3	✗	2xhost	1xhost	DC 5V 3A	✗	✗
ASUS Tinker Board [24]	Cortex A-17 Quad-core 1.8 GHz	2 GB DDR3	CSI	4xhost	✗	DC 5V 2A	✗	✓

A continuación, explicamos las principales virtudes y flaquezas de cada placa relativas a este proyecto.

- Orange Pi Plus 2E [16]: Es una computadora de código abierto de placa única que puede ejecutar Android 4.4, Ubuntu, Debian Image que, además, cuenta con WiFi, por lo que se podría adaptar al tipo de placa que se busca.
- NanoPi Neo plus 2 [17]: Es una placa ARM que soporta WiFi y entrada de video mediante USB, además de contar con el número de puertos USB necesarios para el proyecto.
- Odroid XU4 [18]: Esta placa puede incorporar sistema WiFi y sistema de video.
- UP Core [19]: Es una computadora que se puede usar como placa única o podemos expandirla. También cuenta con Sistema WiFi y permite entrada de video.
- Odroid C2 [20]: Es una microcomputadora que permite sistemas operativos modernos que se ejecutan en ODROID-C2 como Ubuntu, Android, ARCHLinux, Debian. Cuenta con WiFi con antena como modulo USB de forma opcional y entrada de video mediante USB también opcional, por lo que podría ser una de las opciones a adaptar para el proyecto.
- JaguarBoard [21]: Es una computadora de placa única, la diferencia es que ofrece mejor rendimiento con un alto nivel de escalabilidad y compatibilidad. Ampliamente compatible con Windows 8.1 / 10, Ubuntu, CentOS, Fedora, Debian y Android.
- BeagleBone Blue [22]: Es una computadora todo en uno basada en Linux con WiFi, bluetooth, IMU y barómetro entre otras funcionalidades.
- ROCK64 4K60P [23]: Es una computadora de placa única que destaca por su reducido tamaño.
- ASUS Tinker Board [24]: Es una computadora de placa única en un factor de forma muy pequeño, lo cual no afecta en su rendimiento. Además, es compatible con el protocolo 802.11 b / g / n con antena IPEX actualizable.

En la Tabla 4 se muestra un resumen de las especificaciones a tener en cuenta para el proyecto, además del listado que se ha realizado con las características de cada placa que son más relevantes para el proyecto, con lo que podemos concluir que una buena opción podría ser la placa Odroid XU4 ya que cuenta con 2 USB 3.0 y 1 USB 2.0, lo cual haría que incluso nos quedase uno libre por si en un futuro se deciden implementar nuevas funcionalidades. Además, cuenta tanto con puerto UART para poder transmitir y recibir los comandos de vuelo como con la posibilidad de tener conexión WiFi. El único aspecto negativo que podemos destacar de esta placa es que no tiene entrada de video, pero como en principio se planea utilizar una cámara de mayor calidad con entrada USB no sería una gran desventaja.

Una segunda opción sería la placa Orange Pi Plus 2E, que cuenta con un total de 5 puertos USB, tiene puerto UART y posibilidad de conexión WiFi, aunque quizá sea demasiado compleja y además se vayan a desaprovechar bastantes conexiones ya que cuenta con bastantes más puertos de los necesarios en la actualidad.

Como ambas en principio deberían tener un consumo parecido, no descartaremos ninguna hasta concluir el estudio y las diferencias con el resto de las placas.

3.2.2 Placas Arduino

En segundo lugar, se plantea la opción de utilizar un módulo Arduino ya que son de pequeño tamaño y tienen un bajo consumo, por lo que en esta sección se estudiarán las placas de esta familia que sean más relevantes para el proyecto.

Tabla 5: Placas Arduino

Placa	Procesador	Entrada de video	USB 2.0	USB 3.0	Alimentación	UART	WiFi
Yun rev 2 [25]	Atheros AR9331	✗	1xhost	✗	5V	✗	✓
Leonardo [26]	ATmega32u 4	✗	✗	✗	5V	✗	✗
MKR FOX 1200 [27]	SAMD21 Cortex-M0+ 32bit	✗	✓	✗	5V	✓	✗
MKR WAN 1300 [28]	SAMD21 Cortex-M0+ 32bit	✗	✗	✗	3.3V	✓	✗
MKR GSM 1400 [29]	SAMD21 Cortex-M0+ 32bit	✗	✓	✗	3.3V	✓	✗
MKR WIFI 1010 [30]	SAMD21 Cortex-M0+ 32bit	✗	✓	✗	3.3 V	✓	✓
Uno WiFi Rev2 [31]	ATMEGA4 809	✗	✗	✗	5V	✗	✓
MKR NB 1500 [32]	SAMD21 Cortex-M0+ 32bit	✗	✗	✗	3.3V	✓	✗
MKR Vidor 4000 [33]	Intel Cyclone 10CL016	✓	✗	✗	3.3V	✓	✓
MKR 1000 WiFi [34]	SAMD21 Cortex-M0+ 32bit	✗	✗	✗	5V	✓	✗

A continuación, se detallará una lista con las características relevantes que tienen algunas de las placas mostradas en la Tabla 5:

- Leonardo ETM [26]: Esta placa tiene comunicación USB incorporada, lo que elimina la necesidad de un procesador secundario.
- MKR WAN 1300 [28]: Es una placa que conectividad LoRa / LoRaWAN. El Arduino MKR WAN 1300 puede funcionar con o sin las baterías conectadas y tiene un consumo de energía limitado, además cuenta con la posibilidad de tener una antena de 2dB.
- MKR GSM 1400 [29]: Es una placa que puede ofrecer una solución si buscamos agregar conectividad GSM global, aunque su consumo energético es significativo (superior a 500 mA durante las transmisiones).
- MKR WIFI 1010 [30]: Esta placa tiene como objetivo acelerar y simplificar la creación de prototipos de aplicaciones IoT basadas en WiFi gracias a su bajo consumo de energía.
- Uno WiFi Rev2 [31]: Esta placa es funcionalmente la misma que el Arduino Uno Rev3, pero con la adición de WiFi y algunas otras mejoras. El módulo Wi-Fi es un SoC autónomo con una pila de protocolos TCP / IP integrada que puede proporcionar acceso a una red Wi-Fi o actuar como un punto de acceso.
- MKR NB 1500 [32]: Es una placa que se utiliza para agregar comunicación de banda estrecha. Es una opción para dispositivos en ubicaciones remotas sin

conexión a Internet, o en situaciones en las que no hay energía disponible, por ello cuenta con un "Wireless radio".

- MKR Vidor 4000 [33]: Es una placa que podemos configurar dependiendo de las necesidades, ya que dispone de FPGA, esencialmente podemos crear nuestra propia placa controladora. Incluye una interfaz MKR en la cual todos los pines son manejados tanto por el procesador SAMD21 como por la FPGA. Tiene un conector Mini PCI Express con hasta 25 pines programables por el usuario y la FPGA integrada también se puede utilizar para operaciones DSP de alta velocidad para procesamiento de audio y video, por lo que se podría emplear para implementar un codificador de audio y vídeo específico.
- MKR 1000 WiFi [34]: Es una potente placa que utiliza la funcionalidad del Wi-Fi Shield para agregar conectividad Wi-Fi, también incluye una antena interna PCB.

En el caso de los módulos Arduino, mirando la Tabla 5 podemos observar que para implementar todas las funcionalidades que necesitamos sería necesario emplear al menos dos módulos diferentes que tendríamos que sincronizar entre sí, añadiendo una complejidad significativa al sistema para alcanzar el número necesario de puertos USB y UART. Por ello se descarta totalmente la elección de un módulo Arduino para este proyecto, ya que dos módulos implicarían mucho más consumo y peso del que podemos gestionar.

3.2.3 Placas Toradex

Por último, se estudiarán placas más industriales como las que ofrece la empresa Toradex, ya que parece que pueden servir para aportar más funcionalidades en el futuro además de que son placas profesionales de propósito industrial, por lo que brindan una gran estabilidad.

Tabla 6: Placas Toradex

Placa	Procesador	Memoria RAM	USB 2.0	USB 3.0	UART	Entrada de Video	SD	WiFi
APALIS TK1 [35]	NVIDIA Tegra K1 Cortex-A15	2 GB (64 bits)	1xhost	1xhost 1xOTG	10	✓	✓	✗
APALIS IMX8 [36]	Arm Cortex A35 & Arm Cortex M4	4 GB (64 bits) LPDDR4	2x host / 1x OTG	1x Host	7	✓	✓	✓
APALIS IMX6 [37]	Arm Cortex A9	2 GB (64 bits)	4xhost, 1xOTG	✗	5	✓	✓	✗
APALIS T30 [38]	NVIDIA Tegra 3 Cortex A9	2 GB (32 bits)	2xhost, 1xOTG	✗	5	✓	✓	✗
COLIBRI iMX8X [39]	Cortex-A35 & Arm Cortex-M4	2 GB (32 bits) LPDDR4	1xhost, 1x OTG	✗	5	✓	✓	✓
COLIBRI IMX7 [40]	Dual Core Arm Cortex-A7	256 MB – 1 GB	1xhost, 1xOTG	✗	7	✗	✓	✗
COLIBRI IMX6 [41]	Arm Cortex A9	512 MB	1xhost, 1xOTG	✗	5	✗	✓	✗
COLIBRI IMX6ULL [42]	ARM® Cortex-A7	256-512 MB	1xhost, 1xOTG	✗	8	✗	✓	✓
COLIBRI VF61 [43]	Arm Cortex-A5 & Arm Cortex M4	256 MB	1xhost, 1xOTG	✗	5	✗	✓	✗
COLIBRI VF50 [44]	Arm Cortex A5	128 MB	1xhost, 1xOTG	✗	5	✗	✓	✗
COLIBRI T30 [45]	Arm Cortex A9	1 GB (32 bits)	1xhost, 1xOTG	✗	5	✗	✓	✗
COLIBRI T20 [46]	NVIDIA Tegra 2 Arm Cortex A9	512 MB	1xhost, 1xOTG	✗	5	✗	✓	✗

A continuación, se detallará una lista con las características relevantes que tienen algunas de las placas mostradas en la Tabla 6:

- APALIS TK1 [35]: Es un sistema en un módulo que expone una amplia gama de interfaces avanzadas y de alta velocidad como Gbit Ethernet, PCIe, SATA y USB 3.0 y un Interfaz serial de cámara 2x Quad + 1x Single Lane MIPI CSI-2.
- APALIS IMX8 [36]: Es un sistema en el que los núcleos del microcontrolador Cortex M4 se pueden usar para admitir aplicaciones en tiempo real o permanecer conectados mientras los núcleos principales Cortex-A72 / A53 están apagados. Además, cuenta con WiFi y Bluetooth de banda dual 802.11ac 2x2 MU-MIMO integrados a bordo. Aunque esta parece la mejor opción hasta ahora en el momento del estudio de placas aún no se encontraba en venta.
- APALIS IMX6 [37]: El módulo expone una amplia gama de interfaces industriales que incluyen CAN, UART, I2C, USB, PCIe, SATA... además de un interfaz serial de cámara Quad Lane MIPI CSI-2.

- APALIS T30 [38]: Es una computadora pequeña que tiene una amplia gama de interfaces avanzadas y de alta velocidad como Gbit Ethernet, PCIe, SATA… y un Interfaz serial de la cámara 1x Quad o 2x Dual Lane MIPI CSI-2.
- Colibri iMX8X [39]: Es una computadora pequeña que cuenta con seguridad avanzada y compatibilidad con Bluetooth 5 y WiFi Doble banda 802.11ac 2x2 MU-MIMO. Se encontraba en desarrollo en el momento en el que se seleccionó la placa para el proyecto.
- COLIBRI IMX7 [40]: Colibrí iMX7S y Colibrí iMX7D son sistemas en modulo, sus núcleos ARM Cortex-A7 están altamente optimizados para la eficiencia energética y ofrecen un alto rendimiento. Las características de seguridad avanzadas hacen que esta placa sea una buena opción para dispositivos conectados. Cuenta con un microcontrolador Cortex-M4, esto le permite implementar tareas en tiempo real de muy baja latencia de forma rápida y sencilla.
- COLIBRI IMX6 [41]: Es un módulo de computadora que ofrece una amplia gama de interfaces desde GPIO simples, buses I2C, SPI, CAN y UART estándar de la industria hasta interfaces USB 2.0 de alta velocidad y un bus de memoria externa de 16/32 bits (bus paralelo).
- COLIBRI IMX6ULL [42]: Es el primer módulo de computadora de Toradex que ofrece Wi-Fi y Bluetooth integrados que presenta un núcleo Arm ® Cortex-A7 de bajo consumo de energía. Esta placa aún se encuentra en desarrollo.
- COLIBRI VF61 [43]: Es un módulo de computadora con dos núcleos, pueden ejecutar simultáneamente diferentes sistemas operativos. Arm Cortex-A5 ejecuta un sistema operativo convencional como Linux que admite IU y requisitos de alto rendimiento, mientras que ARM Cortex-M4 ejecuta un RTOS en tiempo real Tareas.
- COLIBRI VF50 [44]: Es un módulo de computadora de tamaño que ofrece una CPU rentable y un rendimiento gráfico con un consumo mínimo de energía
- COLIBRI T30 [45]: Es un módulo de computadora que admite DVFS (conmutación dinámica de voltaje y frecuencia) y regulación térmica, lo que permite que el sistema ajuste continuamente la frecuencia y el voltaje de funcionamiento en respuesta a los cambios en la carga de trabajo y la temperatura para proporcionar un rendimiento óptimo con el menor consumo de energía.
- COLIBRI T20 [46]: Es un módulo de computadora de tamaño SODIMM que ofrece un alto rendimiento gráfico y de CPU con un consumo mínimo de energía.

Una vez analizadas todas las placas del fabricante Toradex, destacamos en especial las especificaciones que ofrecen tanto Colibrí iMX8X como Colibrí iMX6ULL, pero en el momento en el que se hizo este estudio ambas se encontraban en desarrollo y por lo tanto podrían ser una buena opción para mejorar el diseño en el futuro, pero no en el momento del estudio.

Por lo tanto, las dos opciones más viables son APALIS T30 y COLIBRI IMX7, por el mismo motivo que en la primera sección, la Apalis T30 cuenta con muchas funciones y puertos que posiblemente sean desaprovechadas, además de que se necesita un mayor presupuesto. Por lo tanto, la elección dentro de este tipo de placa sería la placa Colibrí IMX7 que se adapta a las necesidades de este proyecto en todo, excepto en que tendremos que encargarnos de adaptar y desplegar los controladores de dispositivo para la conexión Wifi ya que no viene definida de fábrica.

En el caso de Toradex, el sistema se compondrá de dos placas, la placa de computación como tal, más la placa portadora que contiene los puertos e interfaces que se incorporarán a la placa de cómputo, por ello a continuación se pasa a estudiar las opciones de placas portadoras que nos ofrece este fabricante, ya que de elegir alguna de sus placas deberemos

contar también con una portadora compatible y asegurarnos de que se siguen cumpliendo las especificaciones que necesitamos. Aunque en un primer momento podría parecer que tenemos el mismo problema que en el caso de las placas Arduino por tener que incorporar dos módulos, en este caso no es un inconveniente, ya que ambos módulos vienen preparados para comunicarse entre sí y no nos implica más trabajo de sincronización.

Tabla 7: Placas portadoras

PLACA PORTADORA	USB 3.0	USB 2.0	UART	SD	PCI	Interfaz serial cámara
APALIS EVALUATION BOARD [47]	1xhost, 1xOTG	4	3	✓	✓	✗
IXORA CARRIER BOARD [48]	2xhost	1xhost, 1xOTG	3	✓	✓	✓
COLIBRI EVOLUTION BOARD [49]	✗	4xhost, 1xOTG 1xDevice	3	✓	✓	✗
ASTER CARRIER BOARD [50]	✗	1xhost 1xhost/client	3	✓	✗	✗
VIOLA CARRIER BOARD PLUS [51]	✗	2	✓	✓	✓	✗
IRIS CARRIER BOARD [52]	✗	1xhost 1xOTG	3	✓	✗	✗
ORCHID CARRIER BOARD [53]	✗	2xhost 1xdevice	2	✓	✗	✗

A continuación, se detallará una lista con las características relevantes que tienen algunas de las portadoras mostradas en la Tabla 7:

- APALIS EVALUATION BOARD [47]: Es una placa compatible con la familia de módulos Apalis Arm. La placa de evaluación Apalis proporciona un entorno de desarrollo flexible para explorar la funcionalidad y el rendimiento de la familia Apalis Computer-on-Module. Apalis está equipada con una variedad de interfaces de conectividad que incluyen USB 3.0, PCI-Express, Serial ATA, Gigabit Ethernet, I2C, SPI, RS232, RS485 y CAN. Las interfaces multimedia compatibles incluyen audio digital y analógico, HDMI, LVDS, VGA, entre otras.
- IXORA CARRIER BOARD [48]: La placa de soporte es compatible con la familia de módulos Apalis Arm. Las interfaces de alta velocidad incluyen Mini PCI-Express, MicroSD, Gigabit Ethernet y mSATA. La compatibilidad con interfaces industriales comunes, incluidas I2C, SPI, RS232, CAN y GPIO, hace que la placa de soporte Ixora sea perfectamente adecuada para aplicaciones industriales e integradas.
- COLIBRI EVOLUTION BOARD [49]: La placa de evaluación Colibri es compatible con la familia de módulos Colibri Arm. También está equipada con una variedad de interfaces de conectividad que incluyen USB 2.0, 100 Mbit Fast Ethernet, I2C, SPI, RS232, RS485 y CAN. Las interfaces multimedia disponibles incluyen audio analógico, HDMI, LVDS, VGA, entre otras.

- ASTER CARRIER BOARD [50]: Aster es una placa de soporte para la familia Colibri. La placa tiene conectores compatibles con la placa de fabricante Arduino Uno y Raspberry Pi.
- VIOLA CARRIER BOARD PLUS [51]: La placa es compatible con la familia de módulos Colibri Arm. La placa de soporte de Viola está dirigida a aplicaciones sensibles al costo que requieren un factor de forma pequeño. Al compartir un conjunto de características similares a Raspberry Pi y BeagleBoard, Viola está dirigida específicamente a los mercados integrados e industriales con una larga vida útil, disponibilidad y soporte.
- IRIS CARRIER BOARD [52]: La placa portadora es compatible con la familia de módulos Colibri Arm. Las interfaces de comunicación incluyen USB 2.0 host y cliente, y 100 Mbit Fast Ethernet. La compatibilidad con interfaces industriales comunes, incluidas I2C, SPI, RS232, CAN y GPIO, hace que la placa portadora Iris sea perfectamente adecuada para aplicaciones industriales e integradas. Las interfaces multimedia compatibles incluyen DVI, LVDS, VGA, TFT LCD, audio analógico y táctil resistiva.
- ORCHID CARRIER [53]: La placa portadora se diseñó teniendo en cuenta los módulos Colibri PXA. Algunas características de los nuevos módulos Colibri, como USB 2.0 de alta velocidad no son compatibles. Las interfaces de comunicación incluyen host y cliente USB de alta velocidad y Fast Ethernet de 100 Mbit. La compatibilidad con interfaces industriales comunes, incluidas I2C, SPI, RS232 GPIO, hace que la placa portadora Orchid sea adecuada para aplicaciones industriales e integradas.

3.2.4 Conclusiones

Después de estudiar las tres secciones anteriores se toma la decisión de optar por la placa Toradex Colibrí IMX7, ya es la que más se ajusta para el desarrollo de este proyecto. Esto nos lleva a tener que seleccionar una placa portadora de esta familia, por ello y con la finalidad de no eliminar ninguno de los recursos que nos proporciona la placa Colibrí IMX7 se selecciona la portadora Aster Carrier Board, que nos sigue proporcionando tanto los puertos USB necesarios como el puerto UART. Tan solo eliminaría la posibilidad de utilizar la interfaz serial para la cámara, pero se pretende utilizar una cámara conectada mediante USB en el futuro, por lo que no resultaría ningún problema.

3.3 Selección de módulo WiFi

Dado que se ha indicado por parte de la empresa Visiona Ingeniería de Proyectos la necesidad de que el sistema de control de a bordo tenga WiFi incorporado y además que se pueda utilizar a 5 GHz para tener mayor velocidad de transmisión, y la placa seleccionada no cuenta con WiFi, surge la necesidad de incorporar un módulo WiFi que nos proporcione esta funcionalidad. Para ello se estudiarán dos de los módulos que se encuentran en el mercado:

- TL-WN722N [54]: Es un adaptador USB Inalámbrico de Alta Sensibilidad con una velocidad de transmisión de 150MBps que posee una antena desmontable de 4dBi, por lo que se podría cambiar si fuera necesaria más ganancia. Además, tiene la posibilidad de encriptación WPA/WPA2 de la red inalámbrica.
- Archer T2UH [55]: Es un adaptador de alta ganancia de doble banda ya que se puede emplear tanto para 5GHz como 2.4GHz, lo cual ofrece una gran versatilidad. Además, este adaptador viene con la siguiente generación de estándar WiFi -

802.11ac por lo que su velocidad de transmisión puede ser muy elevada contando también con el cifrado de seguridad. También cuenta con una antena externa, por lo que sería posible cambiarla por una de mayor ganancia si fuera necesario en el futuro.

Teniendo en cuenta que el adaptador Archer T2UH es compatible con Linux y además podemos trabajar a 5 GHz, nos decidimos por escoger esta antena para el sistema de vuelo.

3.4 Estudio del funcionamiento de la placa

En la sección anterior se concluyó que la placa Toradex Colibri IMX7 con su respectiva portadora era la que mejor se adapta a las necesidades, por ello en esta sección se detallará la puesta en marcha y configuración de esta y cómo se ha implementado la comunicación con la controladora Pixhawk 2 que se utilizara en el VANT. En la Figura 3-4 y la Figura 3-4 podemos observar cómo queda el montaje de la placa seleccionada y su portadora correspondiente.

Figura 3-4: Montaje de placas delantero

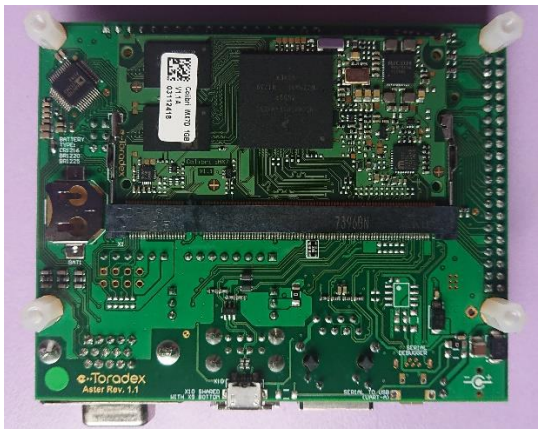

Figura 3-5: Montaje de placas trasero

La placa se vende con el firmware “Toradex Easy Installer” instalado de fábrica, que ofrece una instalación de la distribución Linux Angstrom, lo cual hace que sea sencillo instalar el sistema operativo base en la placa pero nos lleva al problema de que no es la distribución basada en Debian como Raspian y, por lo tanto, esto hará que se tenga que emplear tiempo también en averiguar cuáles son los comandos de consola compatibles con esta distribución de Linux de la placa seleccionada.

Tabla 8: Comandos asociados a distintas distribuciones Linux

Comando en Linux Debian	Comando en Linux Angstrom
apt install	opkg install

Una vez hemos realizado el montaje de las dos placas, enchufado la pantalla, el teclado y el ratón e instalado el sistema operativo mediante la interfaz gráfica empezamos a investigar qué comandos soporta el sistema. En la Tabla 8 podemos ver el ejemplo de cómo cambia el comando de instalación para poder incluir todos los recursos necesarios para la comunicación con la controladora y con el sistema de vuelo. Uno de estos recursos

a instalar es el paquete "`pip`", para la gestión de paquetes de software Python, que es el lenguaje empleado en los módulos software empleados para comunicar con la placa controladora de vuelo y a estación base en tierra.

Se han implementado pequeños códigos en Python, que se adjuntan en el anexo A Códigos empleados para pruebas, para entender el funcionamiento en esta placa particular de los puertos UART y realizar las pruebas de las conexiones TCP/IP entre un ordenador y un programa Python corriendo en la placa. Tras realizar estas pruebas se localizó un código que implementa una pasarela entre un interfaz UART y un servidor TCP/IP [56]. Se adjunta dicho código en el Anexo mencionado anteriormente.

3.5 Compilación del controlador del dispositivo para el adaptador WiFi

Para compilar el controlador de dispositivo (driver en inglés) del adaptador WiFi seleccionado, hemos realizado una búsqueda de diferentes drivers que se encuentran en la red, ya que el desarrollo de un controlador de dispositivo está fuera del alcance de este proyecto. En primer lugar, se prueba con el controlador que facilita la propia web del fabricante [57], pero se detecta que no es un controlador compatible con la placa empleada, como se detalla en el anexo B Compilación del controlador para el adaptador WiFi, por ello nos disponemos a buscar un nuevo controlador que se pueda utilizar.

Seguidamente, se han probado diferentes drivers que se muestran en la Tabla 9. Cada fila corresponde a un driver concreto; en la primera columna se indican las referencias a las paginas donde se han encontrado los códigos, mientras que en la segunda columna se resumen los comandos necesarios para compilarlos.

Tabla 9: Controladores para el adaptador WiFi

[58]	1. `make`
[59]	1. `make` 2. `make install` 3. `cp RT2870STA.dat /etc/Wireless/RT2870STA/RT2870STA.dat` 4. `reboot`
[60]	1. `mkdir ~/src` 2. `cd ~/src` 3. `git clone https://github.com/Myria-de/mt7610u_wifi_sta_v3002_dpo_20130916.git` 4. `make` 5. `make install` 6. `cp RT2870STA.dat /etc/Wireless/RT2870STA/RT2870STA.dat` 7. `reboot`

Ninguno de ellos funciona correctamente, así que se continúa buscando y conseguimos un código referenciado en [61] que nos arroja diferentes errores de compilación. Se detecta que el núcleo proporcionado por Toradex no incorpora las dependencias necesarias para compilar adaptadores WiFi, por lo que se decide recompilar el kernel como se verá en la sección 3.6 Compilación del núcleo del sistema operativo que corre en la placa. Una vez realizado, se reintenta la compilación del driver y tras resolver algunos fallos se compila con éxito el driver.

3.6 Compilación del núcleo del sistema operativo que corre en la placa

Como se vio en la sección 3.5, una vez que intentamos compilar el controlador del adaptador WiFi seleccionado nos damos cuenta de que necesitamos cambiar el sistema operativo de la placa debido a que no es compatible con un núcleo tan antiguo. Por lo tanto, lo primero es intentar cambiar el sistema operativo de la placa buscando una versión más actualizada que nos permita seguir trabajando.

En un primer momento, contamos con el apoyo del servicio técnico del fabricante, aunque este apoyo por su parte no fue eficaz ya que se contestaban a preguntas totalmente distintas a las formuladas en los correos intercambiados. Después de este intento fallido se revisó la documentación proporcionada por Toradex [62], pero ninguno de los intentos de modificar el sistema operativo tuvo éxito.

Por lo tanto, el siguiente paso fue buscar en la página web del fabricante si existen diferentes versiones de núcleo para nuestra placa Colibrí IMX7, ya que el que tenemos no soporta WiFi. Nos damos cuenta de que efectivamente existen numerosas versiones de núcleo e intentamos compilar algunas de ellas que pensamos que pueden ser compatibles tanto con la placa (buscando que no fuese una versión de núcleo obsoleta) como con el adaptador WiFi que tenemos [63], para ello será necesario ajustar la configuración de compilación del kernel para que se puedan compilar las dependencias que WiFi requiere, esto se hará utilizando los comandos que se pueden observar en la Tabla 10 [64], en el ordenador local, esto nos servirá para abrir el menú de configuración del kernel que se muestra en la Figura 3-6, siguiendo la siguiente ruta en el menú: Networking support->Wireless y activando las opciones de WiFi tal como se indica en la propia figura. Una vez configurado el núcleo con las opciones de WiFi, procedemos a compilar el núcleo y, aunque en numerosas ocasiones faltaban librerías o había comandos que la propia placa no admitía, logramos conseguir una versión que compilara sin ningún error, por lo que solo falta verificar si efectivamente también es compatible con el adaptador WiFi.

A continuación, se detallan los comandos para la compilación del núcleo. Para facilitar el proceso y reducir los tiempos de compilación en las pruebas, se ha realizado una compilación cruzada en un ordenador de trabajo (local). Esta compilación cruzada consiste en compilar en una plataforma, por ejemplo, un ordenador tipo PC código que se ejecutará en otra plataforma, como, por ejemplo, un procesador ARM.

Figura 3-6: Configuración del kernel

Tabla 10: Configuración para la compilación del kernel

bitbake -c menuconfig virtual/kernel
bitbake -f -c compile virtual/kernel
bitbake angstrom-lxde-image

Tabla 11: Clonar repositorio de Git

1	mkdir toradex
2	cd toradex
3	git clone -b toradex_4.1-2.0.x-imx git://git.toradex.com/linux-toradex.git

Tabla 12: Instalar GCC linaro para compilación cruzada

1	sudo apt-get install lzop
2	wget -c https://releases.linaro.org/components/toolchain/binaries/6.2-2016.11/arm-linux-gnueabihf/gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf.tar.xz
3	tar xvf gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf.tar.xz
4	ln -s gcc-linaro-6.2.1-2016.11-x86_64_arm-linux-gnueabihf gcc-linaro

Tabla 13: Compilar núcleo

1	export ARCH=arm
2	export PATH=~/toradex/gcc-linaro/bin/:$PATH
3	export CROSS_COMPILE=arm-linux-gnueabihf-
4	cd linux-toradex/
5	make clean
6	make colibri_imx7_defconfig
7	make -j3 2>&1 | tee build.log

En la Tabla 11 se muestran los comandos empleados para clonar el repositorio Git que vamos a utilizar para la compilación del núcleo. En ella el comando de la línea 1 crea la carpeta donde se almacenará el código fuente, el comando de la línea 2 se utiliza para cambiarnos a este directorio que acabamos de crear y el comando de la línea 3 se emplea para el clonado del repositorio.

En la Tabla 12 se muestra cómo se instala el compilador GCC linaro que permite realizar compilaciones cruzadas. En el comando de la línea 1 se instala lzop, que es un programa de compresión rápida necesario para la compilación y empaquetado del núcleo, el comando de la línea 2 se utiliza para descargar el paquete binario para instalar GCC linaro, la línea 3 indica cual es el comando necesario para descomprimir el archivo descargado, que una vez

descomprimido se utilizara en el comando de la línea 4 para crear un enlace simbólico para facilitar la ejecución del compilador.

En la Tabla 13 vemos como compilar el núcleo para que genere los ficheros de configuración y asegurar que compila bien. Para ello utilizamos los comandos de las líneas 1, 2 y 3 para exportar las variables de entorno dinámicas necesarias para configurar la compilación del núcleo, el comando de la línea 4 para cambiar de directorio, el comando de la línea 5 para limpiar los archivos temporales de compilaciones anteriores, si los hubiera, el comando de la línea 6 para crear los ficheros de configuración y el comando de la línea 7 para ejecutar el compilador.

Una vez hemos compilado en el ordenador local y comprobado que todos los parámetros son correctos, en la placa se hace la compilación final para asegurarnos de que disponemos de todas las dependencias y directorios. En este momento, ya tendremos una versión del núcleo compilada y compatible con el controlador del adaptador WiFi que queremos utilizar.

A continuación, se muestra una lista de los errores que nos encontramos durante la compilación:

- *make[1]: Entering directory '/lib/modules/4.1.44-2.7.4+gb1555bf/build'*
  *make[1]: *** No rule to make target 'modules'. Stop.*
  make[1]: Leaving directory '/lib/modules/4.1.44-2.7.4+gb1555bf/build'
  *make: *** [Makefile:370: modules] Error 2*
- *make[1]: Entering directory '/lib/modules/4.1.44-2.7.4+gb1555bf/build'*
  /bin/sh: ./scripts/gcc-goto.sh: No such file or directory
  Makefile:787: scripts/Makefile.extrawarn: No such file or directory
  *make[1]: *** No rule to make target 'scripts/Makefile.extrawarn'. Stop.*
  make[1]: Leaving directory '/lib/modules/4.1.44-2.7.4+gb1555bf/build'
  *make: *** [Makefile:404: LINUX] Error 2*
- *make[1]: Entering directory '/lib/modules/4.1.44-2.7.4+gb1555bf/build'*
  make: Entering an unknown directory
  *make: *** empty variable name. Stop.*
  make: Leaving an unknown directory
  *make[1]: *** [Makefile:1387: _module_/home/root/Desktop/mt7610u_wifi_sta_v3002_dpo_20130916/os/linux] Error 2*

3.7 Conclusiones

En este capítulo hemos visto como seleccionamos la placa que más nos conviene para nuestro proyecto, al igual que con el módulo WiFi que queremos incluir. Una vez seleccionados los dos componentes, hemos estudiado cómo funciona la placa adquirida y compilado el controlador del dispositivo WiFi y el núcleo del sistema que corre en la placa. Una vez compilados el controlador y el núcleo, se intenta configurar el módulo WiFi de la forma en la que se facilita en el manual del fabricante que se encuentra referenciado [65]. Sin embargo, nos encontramos con el problema de que no es compatible esta configuración con el controlador utilizado. Por ello la configuración de este módulo se deja para trabajo futuro.

4 Integración, pruebas y resultados

En el capítulo anterior se ha descrito la prueba de concepto realizada con la Raspberry Pi 3 y la controladora PixHawk 2, se ha realizado una comparativa de placas y adaptadores WiFi-disponibles y se ha explicado el proceso de despliegue con la nueva placa seleccionada, en este caso la placa Toradex Colibri IMX7. En este capítulo, se documentan las pruebas realizadas con cada una de las dos placas, así como los resultados obtenidos. Para ello dividiremos el capítulo en dos secciones, una por cada placa.

4.1 Pruebas con Raspberry Pi 3

En primer lugar, se describirán los pasos que se han seguido para conectar la Raspberry Pi 3 tanto al ordenador en base como a la controladora de vuelo Pixhawk 2.

Tabla 14: Comandos para pruebas con Raspberry Pi 3

1	`ssh pi@192.168.0.27`
2	`sudo mavproxy.py --master=/dev/serial0 --baudrate 115200 --out 192.168.1.5:14550 --aircraft MyCopter`
3	`raspivid -o - -t 0 -n -w 600 -h 400 -fps 12 | cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8000/}' :demux=h264`

En la Tabla 14**¡Error! No se encuentra el origen de la referencia.** se muestran todos los comandos que necesitaremos a lo largo de esta sección para realizar las pruebas de funcionamiento con la Raspberry Pi 3. Para ello, en primer lugar se utiliza el comando de la línea 1 en la Tabla 14 para conectar a la Raspberry Pi 3, siendo "`pi`" el nombre de usuario y "`192.168.0.27`" la dirección IP de la Raspberry. La contraseña que viene por defecto configurada en la Raspberry es "`raspberry`".

Una vez que hemos accedido a la Raspberry, ejecutamos el comando de la línea 2 en la Raspberry para enlazar la controladora y la Raspberry con la aplicación QGroundControl, siendo "`mavproxy.py`" el nombre del fichero de código Python que se utiliza para establecer la comunicación entre los dispositivos, "`/dev/serial0`" el dispositivo de Linux que apunta al puerto de la Raspberry que se está utilizando para la comunicación, "`115200`" la tasa de baudios empleada, "`192.168.1.5`" es la dirección IP del PC donde se está ejecutando el programa de vuelo y "`14550`" el número de puerto para establecer la comunicación.

A continuación, abrimos el programa QGroundControl y lo enlazamos con el sistema y configuramos la recepción de la señal de video empleando el protocolo RTSP. Para ello, definimos en QGroundControl la dirección URL "`rtsp://192.168.0.27:8000`" en la configuración de retransmisión de video para que pueda conectarse a la Raspberry, siendo en este caso 192.168.0.27 la dirección IP de la Raspberry y 8000 el puerto al que nos conectamos. Además, también es posible transmitir el video a través de VLC en lugar de QGroundControl.

Para ejecutar el video a través de VLC se debe ejecutar el comando descrito en la línea 3 de la Tabla 14 en la terminal donde hemos iniciado la sesión de la Raspberry. Para la configuración de transmisión del video mediante VLC, se utilizará de nuevo la URL "`rtsp://192.168.0.27:8000/`" en el propio programa VLC, en la sección de red.

En la Figura 4-1, se muestra el resultado obtenido con la cámara de la Raspberry Pi 3. Por un lado, tenemos el resultado que se muestra por consola donde se observa el envío de datos para la retransmisión y por otro una imagen capturada de lo que se está

retransmitiendo en ese momento. En la captura se muestra la imagen de lo que se está viendo por consola con la finalidad de demostrar que efectivamente está funcionando la retransmisión, así como poder medir la latencia extremo-a-extremo con la que nos encontramos, que en este caso podemos ver que es de aproximadamente 6 segundos, un valor nada aceptable para un dron que debe controlar un operador en tiempo real y que nos lleva a concluir que emplear una Raspberry Pi, aunque es relativamente sencillo, no parece una solución eficaz para un dron.

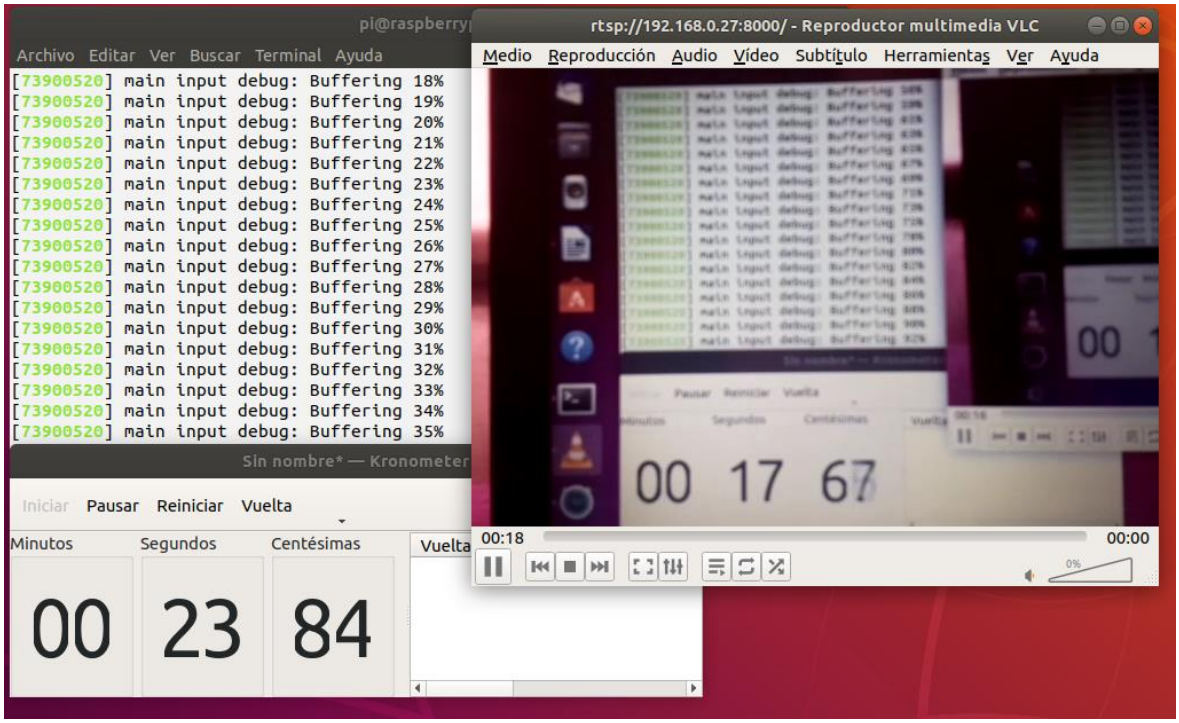

Figura 4-1:Muestra de imagen tomada con la cámara de Raspberry Pi 3

4.2 Pruebas con Toradex IMX7

En segundo lugar, se mostrará cómo se ha utilizado el código referenciado [56] para conectar la placa Toradex Colibri IMX7 con el programa de vuelo de QGroundControl de manera que se puedan enviar los comandos de vuelo con este nuevo ordenador de a bordo y recibir la telemetría de la controladora de vuelo.

Tabla 15: Comandos para pruebas con Toradex IMX7

1	`ssh root@192.168.0.28`
2	`Serial2Net.py /dev/ttymxc0 -P 5760`

En la Tabla 15 se muestran todos los comandos que necesitaremos a lo largo de esta sección para realizar las pruebas de funcionamiento con la placa Toradex Colibri IMX7. En primer lugar, utilizaremos el comando de la línea 1 para acceder a la placa Toradex Colibri IMX7 a través de SSH, como hicimos con la Raspberry Pi; emplearemos el usuario "root" que por defecto no tiene contraseña. Para realizar estas pruebas se utilizará una conexión Ethernet cableada en lugar de WiFi, por el problema que nos hemos encontrado a lo largo del proyecto con el adaptador WiFi descrito en la sección 3.5.

Una vez que nos hemos conectado a la placa, introduciremos el comando de la línea 2 en la terminal de la placa Toradex, para ejecutar el código Serial2Net.py que sirve para hacer

de pasarela entre la interfaz UART "/dev/ttymxc0" donde está conectada la controladora de vuelo PixHawk 2 y un servidor TCP/IP que escuchará conexiones entrantes en el puerto 5760.

En la Figura 4-2 se muestra cómo configurar la conexión TCP en QGroundControl utilizando la dirección IP de la placa Toradex "192.168.0.28" y el puerto 5760 donde escucha el servidor TCP. Seguidamente, en la Figura 4-3 podemos observar que el enlace se ha establecido correctamente. Por una parte, vemos el enlace establecido en QGroundControl contra la dirección IP y el puerto seleccionados, en este caso 192.168.0.28:5760. De haber un error, en ese mensaje se detallaría información relativa al problema. Y por otro lado tenemos la pasarela corriendo en la placa Toradex que conecta a la placa mediante el puerto UART corriendo y con el enlace MAVLink operativo.

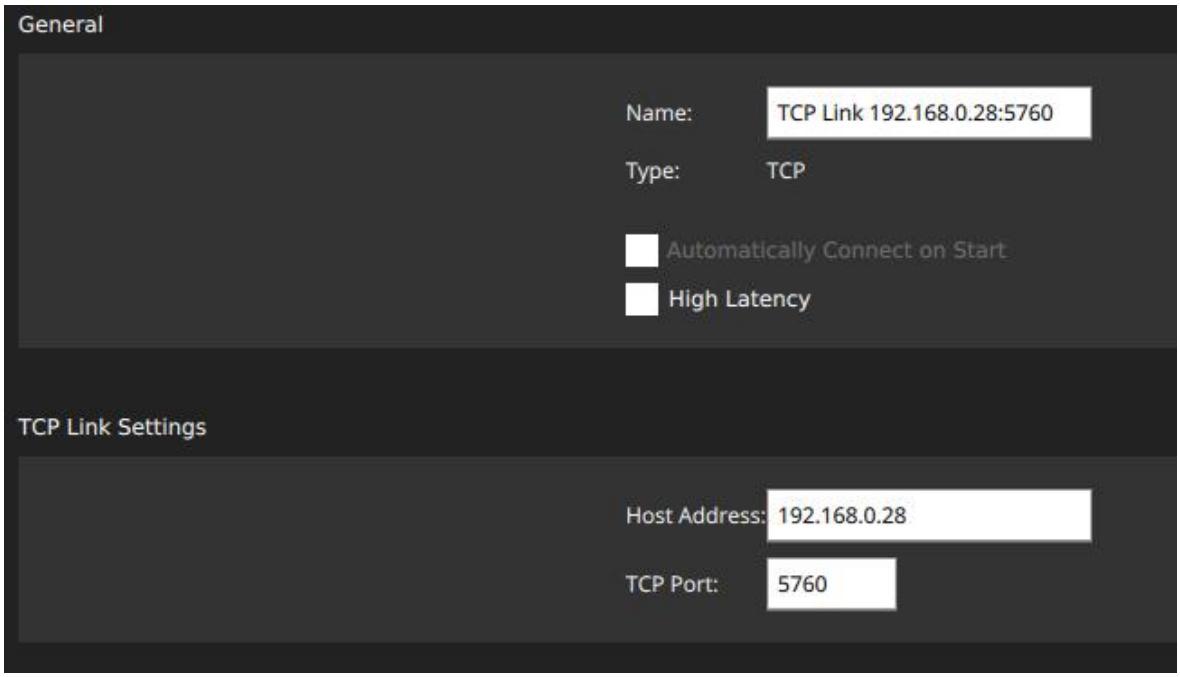

Figura 4-2: Configuración de conexión en QGroundControl

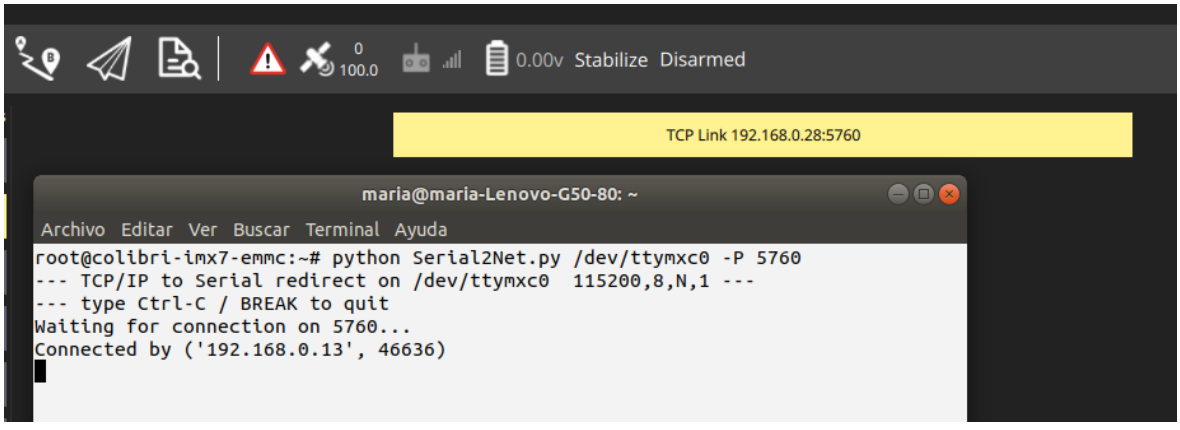

Figura 4-3: Prueba de conexión exitosa entre QGroundControl y la placa

¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.¡Error! No se encuentra el origen de la referencia.

5 Conclusiones y trabajo futuro

5.1 Conclusiones

Para finalizar este proyecto, hablaré de las conclusiones y resultados obtenidos en este trabajo, así como los diferentes problemas y obstáculos a los que me he enfrentado.

En este trabajo se ha abordado el despliegue de un radio enlace que integra tanto la comunicación entre la placa controladora y la estación base, como la retransmisión de video en tiempo real, ambos, a través de un enlace basado en WiFi.

En el capítulo 1 de Introducción se presentó la motivación para realizar este proyecto, que se trataba de desarrollar un canal de comunicación entre el dron y una estación base en tierra.

El capítulo 2 de Estado del arte se presentaron las tecnologías más relevantes para el desarrollo del proyecto.

El capítulo 3 de Diseño y desarrollo, explicamos cuales eran los componentes que necesitaba nuestro sistema, así como los diferentes estudios y comparativas que se llevaron a cabo para seleccionar la mejor placa y módulo WiFi para que se pudiera realizar este proyecto a nivel industrial. Para ello, primero se ha realizado una prueba de concepto empleando una Raspberry Pi 3 conectada a una controladora Pixhawk 2. Un resultado importante de este proyecto es observar que la latencia extremo-a-extremo introducida por la Raspberry en el procesado de la señal de vídeo, alrededor de 6 segundos, hace inviable su empleo en un dron industrial que requiera de una retransmisión de vídeo en tiempo real.

Dado que la solución empleando la Raspberry Pi sufre de falta de estabilidad de funcionamiento e introduce una latencia demasiado elevada en la señal de vídeo, se ha decidido migrar el ordenador de a bordo a una solución viable a nivel industrial. Se ha realizado una comparativa de soluciones tecnológicas disponibles, seleccionando finalmente la placa Toradex IMX7 con su correspondiente placa portadora. Una vez que seleccionamos la placa, se ha hecho la puesta en marcha de ella, para poder finalmente trabajar con ella. Lo que nos lleva a tener que compilar tanto el núcleo de esta como el controlador del adaptador WiFi, ya que la placa Toradex no lo soporta de forma nativa.

Por último, en el capítulo 4 de Integración, pruebas y resultados, se muestran diferentes pruebas que se han llevado a cabo a lo largo del proyecto, vemos como una vez que se han compilado estas dos partes del sistema con éxito, se realizan las pruebas necesarias y vemos que hemos conseguido que el enlace funcione correctamente, así como la retransmisión de video, dejando para trabajo a futuro la configuración final del adaptador WiFi.

Para la realización de este proyecto, han sido de mucha utilidad algunas de las asignaturas cursadas a lo largo de la carrera de Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación. Entre ellas cabe destacar Arquitectura de redes 1 y Arquitectura de redes 2 que han proporcionado los conocimientos necesarios en el campo de las redes de computadores y los diferentes protocolos de red utilizados, en este caso para la comunicación del ordenador de a bordo del dron con la base en tierra que tenemos en el PC local. También, han sido de gran utilidad los conocimientos adquiridos en la asignatura de Redes Multimedia, ya que en ella se presentaban los conceptos de

retransmisión de video en tiempo real y se presentaban herramientas como VLC empleadas en el proyecto para la reproducción de la señal de video retransmitida por el dron, así como los conocimientos requeridos para saber analizar sus resultados. Por último, también ha sido de utilidad la asignatura de Sistemas Electrónicos Digitales, donde se imparten conocimientos de electrónica e interfaces digitales como el UART empleados en el trabajo para conectar con la controladora de vuelo PixHawk.

5.2 Trabajo futuro

También es importante destacar las posibles líneas de desarrollo que se pueden seguir para continuar el proyecto y mejorarlo.

En primer lugar, han quedado abiertos algunos problemas como puede ser la instalación del núcleo que se consiguió compilar. Una vez resuelto este problema, se debería verificar la estabilidad de la placa Toradex con el nuevo núcleo, y recompilar e instalar el controlador del adaptador WiFi. Como último recurso en cuanto al controlador, cabría la posibilidad de extender algún controlador de código abierto disponible. Una vez configurado el adaptador WiFi se deberán repetir las pruebas de funcionamiento.

Asumiendo que el despliegue del núcleo y el controlador del adaptador WiFi se han desplegado correctamente en el micro-ordenador industrial, el siguiente paso sería conectar la cámara al puerto USB y verificar que la placa Toradex la reconoce, que las aplicaciones de procesado de video se pueden instalar y que es capaz de retransmitir la señal de vídeo empleando las aplicaciones Python usadas para la prueba de concepto con la Raspberry Pi.

Por último, cuando se haya conseguido embarcar la placa con la controladora de vuelo y la cámara en el propio dron y se haya comprobado que todo funciona correctamente, se debería re-verificar la prueba de latencia para comprobar que devuelva un valor aceptable. De no ser así, Se podría evaluar el uso de placas industriales más avanzadas que incluyeran FPGAs y GPUs para implementar codificadores de vídeo específicos para este producto, así como la incorporación de nuevos sensores.

Referencias

1. IEEE 802.11, The Working Group Setting the Standards for Wireless LANs. Recuperado de http://www.ieee802.org/11/
2. Colaboradores de Wikipedia. (2019, 10 diciembre). IEEE 802.11n. Recuperado de https://es.wikipedia.org/wiki/IEEE_802.11n
3. colaboradores de Wikipedia. (2019a, noviembre 26). IEEE 802.11ac. Recuperado de https://es.wikipedia.org/wiki/IEEE_802.11ac
4. Grabación aérea de eventos deportivos con drones - Skydron. (2020, febrero 22). Recuperado de https://www.skydron.es/grabacion-video-aereo-deportes/
5. C. (2017, agosto 24). La entrega de paquetes con drones ya es una realidad. Recuperado de https://cnnespanol.cnn.com/video/cnnee-portafolio-entregas-con-drones/
6. Almendros, A. G. (2019, diciembre 16). Utilización de los drones en gestión de emergencias. Recuperado de http://www.hispaviacion.es/utilizacion-de-los-drones-en-gestion-de-emergencias/
7. Drones contra incendios: los nuevos guardabosques. (2020, enero 9). Recuperado de https://blogthinkbig.com/peoplefirst/drones-contra-incendios
8. García, E. R. (2019, julio 24). Drones que se cargan mientras vuelan, lo nuevo del ejército de EEUU. Recuperado de https://www.elespanol.com/omicrono/tecnologia/20180906/drones-cargan-vuelan-nuevo-ejercito-eeuu/335967788_0.html
9. Hex Cube (Pixhawk 2) · PX4 v1.9.0 User Guide. (s.f.). Recuperado 22 febrero, 2020, de https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk-2.html
10. Mission Planner Home — Mission Planner documentation. (2020). Recuperado 2 Julio 2020, de https://ardupilot.org/planner/
11. QGC - QGroundControl - Drone Control. (2020). Recuperado 2 Julio 2020, de http://qgroundcontrol.com/
12. Guía para desarrolladores de MAVLink. . Recuperado el 17 de mayo de 2020 https://mavlink.io/en/
13. Dialects, Recuperado el 17 de mayo de 2020 https://mavlink.io/en/messages/
14. Configurando su Raspberry Pi. (s.f.). Recuperado 22 febrero, 2020, de https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up
15. Installing operating system images - Raspberry Pi Documentation. Recuperado de https://www.raspberrypi.org/documentation/installation/installing-images/
16. Plus2e - Orange Pi Plus 2E. (s.f.). Recuperado 22 febrero, 2020, de http://www.orangepi.org/orangepiplus2e/
17. NanoPi NEO Plus2. (s.f.). Recuperado 22 febrero, 2020, de http://nanopi.io/nanopi-neo-plus2.html
18. Tarjeta Odroid-XU4 con Heat Sink - KUBII. (s.f.). Recuperado 22 febrero, 2020, de https://www.kubii.es/odroid-kubii/2101-tarjeta-odroid-xu4-con-heat-sink-kubii-3272496009844.html
19. UP Core - UP Board. (s.f.). Recuperado 22 febrero, 2020, de https://up-shop.org/up-core/271-up-core.html
20. Odroid C2 - 64-bit quad-core Single Board Computer. (s.f.). Recuperado 22 febrero, 2020, de https://www.odroid.co.uk/hardkernel-odroid-c2-board

21. Jaguarboard : Jaguar One. (s.f.). Recuperado 22 febrero, 2020, de http://www.jaguarboard.org/index.php/com_virtuemart_menu_configuration/products/buy/jaguarboard/207/jaguarboard-detail.html
22. BeagleBoard.org - blue. (s.f.). Recuperado 22 febrero, 2020, de https://beagleboard.org/blue
23. ROCK64. (s.f.). Recuperado 22 febrero, 2020, de https://wiki.pine64.org/index.php/ROCK64
24. Tinker Board | Single-board Computer | ASUS España. (s.f.). Recuperado 22 febrero, 2020, de https://www.asus.com/es/Single-Board-Computer/Tinker-Board/
25. Arduino Yún Rev 2 | Arduino Official Store. (s.f.). Recuperado 22 febrero, 2020, de https://store.arduino.cc/arduino-yun-rev-2
26. Arduino - ArduinoBoardLeonardo. (s.f.). Recuperado 22 febrero, 2020, de https://www.arduino.cc/en/Main/Arduino_BoardLeonardo
27. Arduino MKR FOX 1200. (s.f.). Recuperado 22 febrero, 2020, de https://store.arduino.cc/arduino-mkr-fox-1200-1408
28. Arduino MKR WAN 1300 (LoRa connectivity). (s.f.). Recuperado 22 febrero, 2020, de https://store.arduino.cc/arduino-mkr-wan-1300-lora-connectivity-1414
29. Arduino MKR GSM 1400. (s.f.). Recuperado 22 febrero, 2020, de https://store.arduino.cc/arduino-mkr-gsm-1400-1415
30. Arduino MKR WiFi 1010 | Arduino Official Store. (s.f.). Recuperado 22 febrero, 2020, de https://store.arduino.cc/arduino-mkr-wifi-1010
31. ARDUINO UNO WiFi REV2 | Arduino Official Store. (s.f.). Recuperado 22 febrero, 2020, de https://store.arduino.cc/arduino-uno-wifi-rev2
32. Arduino MKR NB 1500. (s.f.). Recuperado 22 febrero, 2020, de https://store.arduino.cc/arduino-mkr-nb-1500-1413
33. Arduino MKR Vidor 4000 | Arduino Official Store. (s.f.). Recuperado 22 febrero, 2020, de https://store.arduino.cc/arduino-mkr-vidor-4000
34. Arduino MKR1000 WiFi | Arduino Official Store. (s.f.). Recuperado 22 febrero, 2020, de https://store.arduino.cc/arduino-mkr1000-wifi
35. NVIDIA Tegra K1 System/Computer on Module - Apalis TK1 SoM. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/computer-on-modules/apalis-arm-family/nvidia-tegra-k1
36. NXP i.MX 8QM Computer on Module - Apalis iMX8. Recuperado de https://www.toradex.com/computer-on-modules/apalis-arm-family/nxp-imx-8
37. NXP/Freescale i.MX 6 Arm based Computer on Module - Apalis iMX6. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/computer-on-modules/apalis-arm-family/nxp-freescale-imx-6
38. NVIDIA Tegra 3 Computer on Module - Apalis T30. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/computer-on-modules/apalis-arm-family/nvidia-tegra-3
39. NXP i.MX 8X Computer on Module - Colibri iMX8QXP - Arm Cortex A35. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/computer-on-modules/colibri-arm-family/nxp-imx-8x
40. NXP i.MX 7 - Arm Computer/System on Module. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/computer-on-modules/colibri-arm-family/nxp-freescale-imx7
41. NXP/Freescale i.MX 6 based Arm Computer Module - Colibri iMX6. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/computer-on-modules/colibri-arm-family/nxp-freescale-imx6

42. NXP i.MX 6ULL - Arm Computer/System on Module iMX6ULL. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/computer-on-modules/colibri-arm-family/nxp-imx6ull
43. NXP/Freescale Vybrid VF6xx Computer on Module - Arm Cortex A5/M4. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/computer-on-modules/colibri-arm-family/nxp-freescale-vybrid-vf6xx
44. NXP/Freescale Vybrid VF5xx Computer on Module - Colibri VF50. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/computer-on-modules/colibri-arm-family/nxp-freescale-vybrid-vf5xx
45. NVIDIA Tegra 3 Computer on Module - Colibri T30. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/computer-on-modules/colibri-arm-family/nvidia-tegra-3
46. NVIDIA Tegra 2 Computer on Module - Colibri T20. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/computer-on-modules/colibri-arm-family/nvidia-tegra-2
47. Evaluation and Development on Apalis modules. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/products/carrier-board/apalis-evaluation-board
48. Ixora Carrier Board. (s.f.). Recuperado 22 febrero, 2020, de https://developer.toradex.com/products/ixora-carrier-board
49. Evaluation and Development on Colibri modules. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/products/carrier-board/colibri-evaluation-board
50. Arduino, Raspberry Pi compatible. Compact - Aster Carrier Board. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/products/carrier-boards/aster-carrier-board
51. Very small size. Low cost. Mass production on Colibri module. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.cn/en/products/carrier-board/viola-carrier-board
52. Compact. Development, mass production on Colibri modules. (s.f.). Recuperado 22 febrero, 2020, de https://www.toradex.com/products/carrier-board/iris-carrier-board
53. Orchid Carrier Board | Toradex Carrier Boards. (s.f.). Recuperado 22 febrero, 2020, de https://developer.toradex.com/products/legacy/carrier-boards/orchid-carrier-board
54. Download for TL-WN722N | TP-Link Iberia. TP-Link. https://www.tp-link.com/es/support/download/tl-wn722n/
55. Archer T2UH | Adaptador USB inalámbrico de alta ganancia de doble banda AC600 | TP-Link Iberia. Tp-Link. https://www.tp-link.com/es/home-networking/adapter/archer-t2uh/
56. Shaoyu (s.f.). pyserial/pyserial. Recuperado 22 febrero, 2020, de https://github.com/pyserial/pyserial/blob/master/examples/tcp_serial_redirect.py
57. Download for Archer T2UH | TP-Link Iberia. Recuperado de https://www.tp-link.com/es/support/download/archer-t2uh/#Driver
58. Build software better, together. Recuperado de https://github.com/ulli-kroll/mt7610u
59. Repositorio GitHub chenhaiq/mt7610u_wifi_sta_v3002_dpo_20130916. Recuperado de https://github.com/chenhaiq/mt7610u_wifi_sta_v3002_dpo_20130916
60. Repositorio GitHub lixz789/mt7610u_wifi_sta_v3002_dpo_20130916. Recuperado de https://github.com/lixz789/mt7610u_wifi_sta_v3002_dpo_20130916
61. Repositorio GitHub xtknight/mt7610u-linksys-ae6000-wifi-fixes. Recuperado de https://github.com/xtknight/mt7610u-linksys-ae6000-wifi-fixes

62. Compiling Linux Kernel - Toradex Community. Recuperado de https://www.toradex.com/community/questions/6114/compiling-linux-kernel.html
63. Kernel Driver Backports Integration. Recuperado de https://developer.toradex.com/knowledge-base/kernel-backports-integration
64. Adding WiFi Driver to the Kernel - Toradex Community. Recuperado de https://www.toradex.com/community/questions/1309/adding-wifi-driver-to-the-kernel.html
65. https://static.tp-link.com/2019/201910/20191018/Installation%20for%20linux%20driver_MTK.PDF

Glosario

ARM	Advanced RISC Machine
CAN	Controller Area Network
CPU	Unidad central de procesamiento
ESC	Electronic Speed Controller
FPGA	Field-programmable gate array
GPIO	General Purpose Input/Output
GPS	Global Positioning System
PC	Personal Computer
RTSP	Real Time Streaming Protocol
SD	Secure Digital
TCP	Protocolo de Control de Transmisión
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
VANT	Vehículo aéreo no tripulado

Anexos

A Códigos empleados para pruebas

A continuación, se adjuntan una serie de códigos escritos en Python que se han utilizado durante este proyecto para ir desarrollando el enlace de comunicación y comprobar que funcionaba correctamente.

Para ello, en primer lugar, se crearon dos pequeños códigos para probar que funcionaba la comunicación mediante un socket TCP que se encargue de mandar mensajes por consola y recibirlos en otro terminal. Estos mensajes escritos por terminal simbolizan los comandos que la controladora mandará al ordenador de a bordo del VANT.

El código que se encarga de mandar los mensajes, en este caso escritos por consola, es el siguiente:

```
import socket, sys

HOST, PORT = "192.168.1.78", 10000
data = " ".join(sys.argv[1:])
print 'data = %s' %data

# create a TCP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
    # connect to server
        sock.connect((HOST, PORT))

    # send data
        sock.sendall(bytes(data + "\n"))

    # receive data back from the server
        received = str(sock.recv(1024))
finally:
    # shut down
        sock.close()

print("Sent:     {}".format(data))
print("Received: {}".format(received))
```

El código que se encargara de recibir estos pequeños mensajes es el siguiente:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import sys
import SocketServer

class MyTCPHandler(SocketServer.StreamRequestHandler):
    def handle(self):
        for line in self.rfile:
            print line
```

```
        print 'end'

def main():
    host = '0.0.0.0'
    port = 10001
    server = SocketServer.TCPServer((host, port), MyTCPHandler)
    server.serve_forever()

if __name__ == '__main__':
    sys.exit(main())
```

También se lista el código empleado para configurar la pasarela UART-a-TCP [56]. Se trata de un código que es capaz de establecer la comunicación entre la controladora de vuelo y la estación base en tierra, como se detalla en la sección Pruebas con Toradex IMX7.

```
# Simple Serial to Network (TCP/IP) redirector
# NOTE: no security measures are implemented. Anyone can remotely
connect
# to this service over the network.
# Only one connection at once is supported. When the connection is
terminated
# it waits for the next connect.
# Reference:
https://github.com/pyserial/pyserial/blob/master/examples/tcp_serial_red
irect.py

# Execution: python Serial2Net.py /dev/ttymxc0

import sys
import socket
import serial
import serial.threaded
import time

class SerialToNet(serial.threaded.Protocol):
    """serial->socket"""

    def __init__(self):
        self.socket = None

    def __call__(self):
        return self

    def data_received(self, data):
        if self.socket is not None:
            self.socket.sendall(data)


if __name__ == '__main__':  # noqa
    import argparse

    parser = argparse.ArgumentParser(
```

```
        description='Simple Serial to Network (TCP/IP) redirector.',
        epilog="""\
NOTE: no security measures are implemented. Anyone can remotely connect
to this service over the network.

Only one connection at once is supported. When the connection is
terminated
it waits for the next connect.
""")

    parser.add_argument(
        'SERIALPORT',
        help="serial port name")

    parser.add_argument(
        'BAUDRATE',
        type=int,
        nargs='?',
        help='set baud rate, default: %(default)s',
        default=115200)

    parser.add_argument(
        '-q', '--quiet',
        action='store_true',
        help='suppress non error messages',
        default=False)

    parser.add_argument(
        '--develop',
        action='store_true',
        help='Development mode, prints Python internals on errors',
        default=False)

    group = parser.add_argument_group('serial port')

    group.add_argument(
        "--bytesize",
        choices=[5, 6, 7, 8],
        type=int,
        help="set bytesize, one of {5 6 7 8}, default: 8",
        default=8)

    group.add_argument(
        "--parity",
        choices=['N', 'E', 'O', 'S', 'M'],
        type=lambda c: c.upper(),
        help="set parity, one of {N E O S M}, default: N",
        default='N')

    group.add_argument(
        "--stopbits",
        choices=[1, 1.5, 2],
        type=float,
```

```
        help="set stopbits, one of {1 1.5 2}, default: 1",
        default=1)

    group.add_argument(
        '--rtscts',
        action='store_true',
        help='enable RTS/CTS flow control (default off)',
        default=False)

    group.add_argument(
        '--xonxoff',
        action='store_true',
        help='enable software flow control (default off)',
        default=False)

    group.add_argument(
        '--rts',
        type=int,
        help='set initial RTS line state (possible values: 0, 1)',
        default=None)

    group.add_argument(
        '--dtr',
        type=int,
        help='set initial DTR line state (possible values: 0, 1)',
        default=None)

    group = parser.add_argument_group('network settings')

    exclusive_group = group.add_mutually_exclusive_group()

    exclusive_group.add_argument(
        '-P', '--localport',
        type=int,
        help='local TCP port',
        default=7777)

    exclusive_group.add_argument(
        '-c', '--client',
        metavar='HOST:PORT',
        help='make the connection as a client, instead of running a
server',
        default=False)

    args = parser.parse_args()

    # connect to serial port
    ser = serial.serial_for_url(args.SERIALPORT, do_not_open=True)
    ser.baudrate = args.BAUDRATE
    ser.bytesize = args.bytesize
    ser.parity = args.parity
    ser.stopbits = args.stopbits
    ser.rtscts = args.rtscts
```

```
    ser.xonxoff = args.xonxoff

    if args.rts is not None:
        ser.rts = args.rts

    if args.dtr is not None:
        ser.dtr = args.dtr

    if not args.quiet:
        sys.stderr.write(
            '--- TCP/IP to Serial redirect on {p.name}
{p.baudrate},{p.bytesize},{p.parity},{p.stopbits} ---\n'
            '--- type Ctrl-C / BREAK to quit\n'.format(p=ser))

    try:
        ser.open()
    except serial.SerialException as e:
        sys.stderr.write('Could not open serial port {}:
{}\n'.format(ser.name, e))
        sys.exit(1)

    ser_to_net = SerialToNet()
    serial_worker = serial.threaded.ReaderThread(ser, ser_to_net)
    serial_worker.start()

    if not args.client:
        srv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        srv.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        srv.bind(('', args.localport))
        srv.listen(1)
    try:
        intentional_exit = False
        while True:
            if args.client:
                host, port = args.client.split(':')
                sys.stderr.write("Opening connection to
{}:{}...\n".format(host, port))
                client_socket = socket.socket()
                try:
                    client_socket.connect((host, int(port)))
                except socket.error as msg:
                    sys.stderr.write('WARNING: {}\n'.format(msg))
                    time.sleep(5)  # intentional delay on reconnection
as client
                    continue
                sys.stderr.write('Connected\n')
                client_socket.setsockopt(socket.IPPROTO_TCP,
socket.TCP_NODELAY, 1)
                #~ client_socket.settimeout(5)
            else:
                sys.stderr.write('Waiting for connection on
{}...\n'.format(args.localport))
                client_socket, addr = srv.accept()
```

```
                sys.stderr.write('Connected by {}\n'.format(addr))
                # More quickly detect bad clients who quit without
closing the
                # connection: After 1 second of idle, start sending TCP
keep-alive
                # packets every 1 second. If 3 consecutive keep-alive
packets
                # fail, assume the client is gone and close the
connection.
                try:
                    client_socket.setsockopt(socket.IPPROTO_TCP,
socket.TCP_KEEPIDLE, 1)
                    client_socket.setsockopt(socket.IPPROTO_TCP,
socket.TCP_KEEPINTVL, 1)
                    client_socket.setsockopt(socket.IPPROTO_TCP,
socket.TCP_KEEPCNT, 3)
                    client_socket.setsockopt(socket.SOL_SOCKET,
socket.SO_KEEPALIVE, 1)
                except AttributeError:
                    pass # XXX not available on windows
                client_socket.setsockopt(socket.IPPROTO_TCP,
socket.TCP_NODELAY, 1)
            try:
                ser_to_net.socket = client_socket
                # enter network <-> serial loop
                while True:
                    try:
                        data = client_socket.recv(1024)
                        if not data:
                            break
                        ser.write(data)                 # get a bunch of
bytes and send them
                    except socket.error as msg:
                        if args.develop:
                            raise
                        sys.stderr.write('ERROR: {}\n'.format(msg))
                        # probably got disconnected
                        break
            except KeyboardInterrupt:
                intentional_exit = True
                raise
            except socket.error as msg:
                if args.develop:
                    raise
                sys.stderr.write('ERROR: {}\n'.format(msg))
            finally:
                ser_to_net.socket = None
                sys.stderr.write('Disconnected\n')
                client_socket.close()
                if args.client and not intentional_exit:
                    time.sleep(5)  # intentional delay on reconnection
as client
    except KeyboardInterrupt:
```

```
    pass

sys.stderr.write('\n--- exit ---\n')
serial_worker.stop()
```

B Compilación del controlador para el adaptador WiFi

A continuación, se detallan los resultados arrojados por pantalla al compilar con el comando `make` el controlador proporcionado por la página web del distribuidor del adaptador WiFi, Tp-link [57]. Como podremos observar al final de los resultados, no se pudo completar correctamente la compilación y por ello en la sección 3.53.5Compilación del controlador del dispositivo para el adaptador WiFi, tendremos que buscar alternativas para la compilación del controlador del módulo WiFi.

```
maria@maria-Lenovo-G50-80: ~/Escritorio$ sudo ssh root@192.168.0.28

Last login: Wed Jul 1 14:53:34 2020 from 192.168.0.13
root@colibri-imx7-emmc:~# cd Desktop/
root@colibri-imx7-emmc:~/Desktop# cd Driver/

root@colibri-imx7-emmc:~/Desktop/Driver# make

make -C UTIL/ osutil

make[1]: Entering directory '/home/root/Desktop/Driver/UTIL'

cp                      -f                      os/linux/Makefile.6.util
/home/root/Desktop/Driver/UTIL/os/linux/Makefile

make             -C             /lib/modules/4.1.44-2.7.4+gb1555bf/build
SUBDIRS=/home/root/Desktop/Driver/UTIL/os/linux modules

make[2]: Entering directory '/lib/modules/4.1.44-2.7.4+gb1555bf/build'
  CC [M]
 /home/root/Desktop/Driver/UTIL/os/linux/../../common/rt_os_util.o
  CC [M]
/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_linux_symb.o
  CC [M]

/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_rbus_pci_util.
o CC [M]

  /home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_usb_util.o

/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_usb_util.c:  In
function 'rausb_fill_bulk_urb':

/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_usb_util.c:425
:68:   warning:   passing   argument   6   of   'usb_fill_bulk_urb'   from
incompatible pointer type [-Wincompatible-pointer-types]
  usb_fill_bulk_urb(urb,   dev,   pipe,   transfer_buffer,   buffer_length,
complete_fn, context);

^~~~~~~~~~
In                  file                  included                  from
/home/root/Desktop/Driver/UTIL/include/os/rt_linux.h:40:0,
                 from
/home/root/Desktop/Driver/UTIL/include/rtmp_os.h:48,
```

```
                 from
/home/root/Desktop/Driver/UTIL/include/rtmp_comm.h:62,
                 from
/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_usb_util.c:18:
include/linux/usb.h:1532:20: note: expected 'usb_complete_t {aka void
(*)(struct urb *)}' but argument is of type 'USB_COMPLETE_HANDLER {aka
void (*)(void *)}'
 static inline void usb_fill_bulk_urb(struct urb *urb,
                    ^~~~~~~~~~~~~~~~~
  CC [M]
/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_linux.o

/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_linux.c:    In
function 'RtmpOsUsDelay':

/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_linux.c:179:8:
warning: unused variable 'i' [-Wunused-variable]
  ULONG i;
        ^
/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_linux.c:    In
function 'RtmpDrvAllRFPrint':
/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_linux.c:2051:3
2:   warning:   passing   argument   2   of   'file_w->f_op->write'   from
incompatible pointer type [-Wincompatible-pointer-types]
    file_w->f_op->write(file_w, pBuf, BufLen, &file_w->f_pos);
                                ^~~~
/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_linux.c:2051:3
2: note: expected 'const char *' but argument is of type 'UINT32 * {aka
unsigned int *}'

/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_linux.c:2036:2
2: warning: unused variable 'macValue' [-Wunused-variable]
  UINT32 macAddr = 0, macValue = 0;
                      ^~~~~~~~

/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_linux.c:2036:9
: warning: unused variable 'macAddr' [-Wunused-variable]
  UINT32 macAddr = 0, macValue = 0;
         ^~~~~~~

/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_linux.c:    In
function 'RtmpOSIRQRelease':
/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_linux.c:2172:2
1: warning: unused variable 'net_dev' [-Wunused-variable]
  struct net_device *net_dev = (struct net_device *)pNetDev;
                     ^~~~~~~
/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_linux.c:    In
function 'RtmpOsFreeSpinLock':

/home/root/Desktop/Driver/UTIL/os/linux/../../os/linux/rt_linux.c:4266:8
: warning: assignment from incompatible pointer type [-Wincompatible-
pointer-types]
  pLock = (OS_NDIS_MINIPORT_TIMER *) (pLockOrg->pContent);
```

```
       ^
  LD [M]  /home/root/Desktop/Driver/UTIL/os/linux/mt7650u_sta_util.o
  Building modules, stage 2.

  MODPOST 1 modules
  CC      /home/root/Desktop/Driver/UTIL/os/linux/mt7650u_sta_util.mod.o
  LD [M]  /home/root/Desktop/Driver/UTIL/os/linux/mt7650u_sta_util.ko
make[2]: Leaving directory '/lib/modules/4.1.44-2.7.4+gb1555bf/build'
make[1]: Leaving directory '/home/root/Desktop/Driver/UTIL'
/bin/sh cp_util.sh

make -C MODULE/ build_tools

make[1]: Entering directory '/home/root/Desktop/Driver/MODULE'

make -C tools

make[2]: Entering directory '/home/root/Desktop/Driver/MODULE/tools'

gcc -g bin2h.c -o bin2h

make[2]: Leaving directory '/home/root/Desktop/Driver/MODULE/tools'
/home/root/Desktop/Driver/MODULE/tools/bin2h

chipset = mt7650u

chipset = mt7630u

chipset = mt7610u

make[1]: Leaving directory '/home/root/Desktop/Driver/MODULE'

make -C MODULE/ osdrv

make[1]: Entering directory '/home/root/Desktop/Driver/MODULE'
cp                          -f                          os/linux/Makefile.6
/home/root/Desktop/Driver/MODULE/os/linux/Makefile

make              -C              /lib/modules/4.1.44-2.7.4+gb1555bf/build
SUBDIRS=/home/root/Desktop/Driver/MODULE/os/linux modules
make[2]: Entering directory '/lib/modules/4.1.44-2.7.4+gb1555bf/build'
  CC [M]
/home/root/Desktop/Driver/MODULE/os/linux/../../os/linux/rt_profile.o

/home/root/Desktop/Driver/MODULE/os/linux/../../os/linux/rt_profile.c:
In function 'announce_802_3_packet':

/home/root/Desktop/Driver/MODULE/os/linux/../../os/linux/rt_profile.c:39
7:16: warning: unused variable 'pAd' [-Wunused-variable]
  RTMP_ADAPTER *pAd = (RTMP_ADAPTER *)pAdSrc;
                ^~~
  CC [M]  /home/root/Desktop/Driver/MODULE/os/linux/../../sta/assoc.o
```

```
  CC [M]  /home/root/Desktop/Driver/MODULE/os/linux/../../sta/auth.o

  CC [M]  /home/root/Desktop/Driver/MODULE/os/linux/../../sta/auth_rsp.o

  CC [M]  /home/root/Desktop/Driver/MODULE/os/linux/../../sta/sync.o

  CC [M]  /home/root/Desktop/Driver/MODULE/os/linux/../../sta/sanity.o
  CC [M]

/home/root/Desktop/Driver/MODULE/os/linux/../../sta/rtmp_data.o

  CC [M]  /home/root/Desktop/Driver/MODULE/os/linux/../../sta/connect.o

  CC [M]  /home/root/Desktop/Driver/MODULE/os/linux/../../sta/wpa.o

  CC [M]  /home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.o
/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:        In
function 'Set_EncrypType_Proc':

/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:910:5:
warning: this 'else' clause does not guard... [-Wmisleading-indentation]
     else
     ^~~~
/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:913:2:
note: ...this statement, but the latter is misleadingly indented as if
it is guarded by the 'else'
  if (pAd->StaCfg.BssType == BSS_ADHOC)
  ^~

/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:        In
function 'RTMPIoctlShow':

/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:7053:85:
error: macro "__DATE__" might prevent reproducible builds [-Werror=date-
time]
 intf(extra, size, "Driver version-%s, %s %s\n", STA_DRIVER_VERSION,
__DATE__, __TIME__ );

^~~~~~~

/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:7053:95:
error: macro "__TIME__" might prevent reproducible builds [-Werror=date-
time]
 , size, "Driver version-%s, %s %s\n", STA_DRIVER_VERSION, __DATE__,
__TIME__ );

^~~~~~~

/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:        In
function 'RtmpIoctl_rt_ioctl_siwfreq':
/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:7307:5:
warning: this 'else' clause does not guard... [-Wmisleading-indentation]
```

```
     else
     ^~~~

/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:7310:2:
note: ...this statement, but the latter is misleadingly indented as if
it is guarded by the 'else'
  return NDIS_STATUS_SUCCESS;
  ^~~~~~

/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:        In
function 'RtmpIoctl_rt_ioctl_giwrate':
/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:9178:5:
warning: this 'if' clause does not guard... [-Wmisleading-indentation]
     if (rate_index >= rate_count)
     ^~

/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:9181:2:
note: ...this statement, but the latter is misleadingly indented as if
it is guarded by the 'if'
  *(ULONG *)pData = ralinkrate[rate_index] * 500000;
  ^

/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:        In
function 'RtmpIoctl_rt_private_get_statistics':
/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:9737:17:
warning: unused variable 'fec_coding' [-Wunused-variable]
    static char *fec_coding[2] = {"bcc", "ldpc"};
                 ^~~~~~~~~~

At top level:
/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.c:9737:17:
warning: 'fec_coding' defined but not used [-Wunused-variable]
cc1: some warnings being treated as errors

make[3]:                 ***                 [scripts/Makefile.build:259:
/home/root/Desktop/Driver/MODULE/os/linux/../../sta/sta_cfg.o] Error 1

make[2]:                       ***                       [Makefile:1387:
_module_/home/root/Desktop/Driver/MODULE/os/linux] Error 2

make[2]: Leaving directory '/lib/modules/4.1.44-2.7.4+gb1555bf/build'
make[1]: *** [Makefile:549: osdrv] Error 2

make[1]: Leaving directory '/home/root/Desktop/Driver/MODULE'
make: *** [Makefile:6: all] Error 2
```